УНИВЕРСИТЕТ
искусственного
интеллекта

Решение задачи регрессии

Решение задачи регрессии

Цель задачи **регрессии** — предсказать значение числовой переменной на основе значений одной или более переменной предикторов (независимых переменных), которые могут быть либо числовыми, либо категориальными.

Примеры задач регрессии:

- предсказание зарплаты по резюме,
- оценка стоимости квартир,
- предсказание курса доллара США к рублю и т. д.

Динамика курса доллара США к рублю (USDTOM_UTS, MOEX)

За последние 10 дней

Дата	Курс	Изменение	
09.08.18	66,1375	0,5900	↑
08.08.18	65,5475	2,0575	↑
07.08.18	63,4900	-0,2500	↓
06.08.18	63,7400	0,4025	↑
03.08.18	63,3375	-0,0675	↓
02.08.18	63,4050	0,3600	↑
01.08.18	63,0450	0,5425	↑
31.07.18	62,5025	0,2650	↑
30.07.18	62,2375	-0,5600	↓
27.07.18	62,7975	-0,1575	↓

Решение задачи регрессии

Регрессия нейронной сети (*RNN*) имеет единственный выходной нейрон, который хранит спрогнозированное значение зависимой числовой переменной.

Если мы, например, имеем дело с такой задачей регрессии, как предсказание зарплаты человека по резюме, то на выходе сети мы хотим получить число, т. е. какую зарплату прогнозирует сеть. При обучении такой сети в x_train мы будем подавать разные данные из резюме (возраст, пол, стаж, опыт и т. д.), а в y_train будем подавать зарплату (число) по этим самым готовым резюме. Доказано, что сеть будет лучше работать, если эти данные подавать нормированными.

Нормирование данных

Исходные значения признаков могут изменяться в очень большом диапазоне и отличаться друг от друга на несколько порядков. Например, в одном входном векторе присутствует информация о возрасте и доходе клиента. Так как возраст будет иметь примерный диапазон от 18 до 60, а зарплата от 100 до 1 000 000, то результат работы сети будет не совсем приемлемым, поэтому нужно привести эти данные к более рациональной форме.

После нормализации все числовые значения входных признаков будут приведены к одинаковой области их изменения – некоторому узкому диапазону. Это позволит свести их в одной модели и обеспечит корректную работу вычислительных алгоритмов.

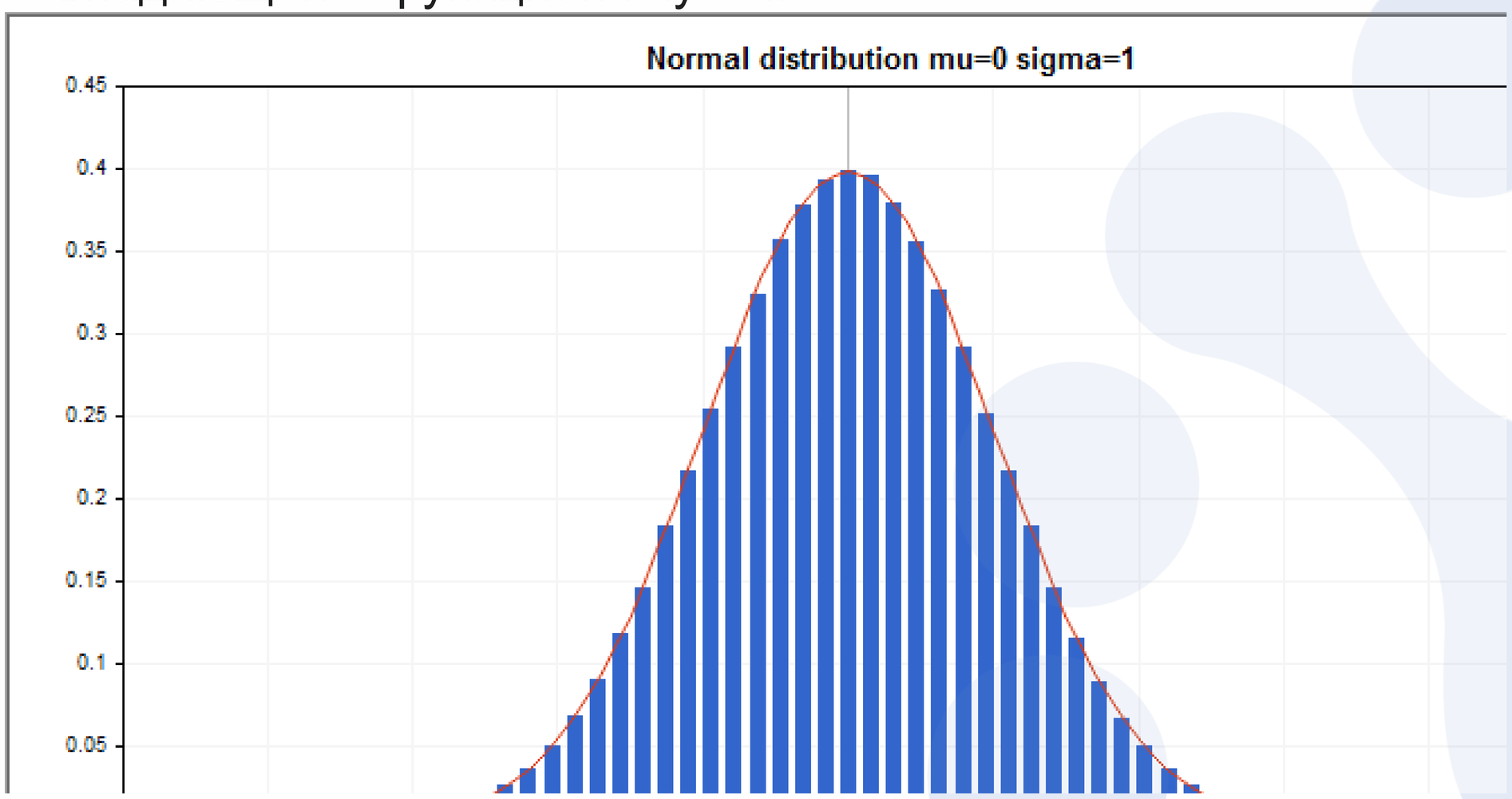
Если данные оставить без нормирования, то их диапазон может быть [-∞, +∞].

Возможные функции активации (с таким диапазоном), которые мы можем использовать на выходном слое нашей модели — ***linear()*** и ***relu(),*** если значения находятся в интервале от 0 до +∞.

Методы нормализации данных

1. Нормальное распределение.

Нормальное распределение, также называемое Гауссовским **распределением** — это распределение вероятностей, которое в одномерном случае задается функцией плотности вероятности, совпадающей с функцией Гаусса.



Нормирование данных

Распределение происходит на основе среднего значения и стандартного отклонения.

На графике показано нормальное распределение от -4 до 4.

В этом случае функция активации выходного слоя должна быть ***linear()***, так как linear() имеет диапазон [-∞, +∞].

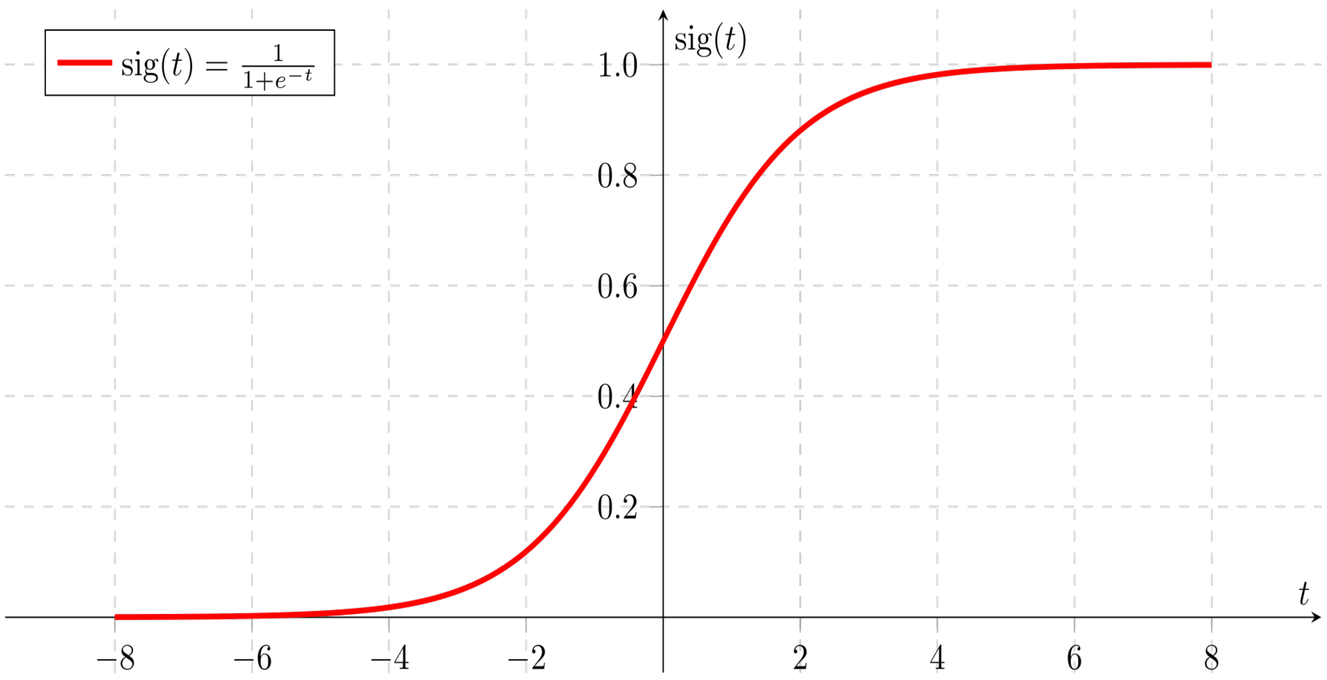
2. MinMax распределение.

MinMax — линейное преобразование данных в диапазоне [0..1], где минимальное и максимальное масштабируемое значение равно 0 и 1 соответственно.

Преобразование данных в интервал [0..1] происходит по следующей формуле:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

При таком распределении функция активации выходного слоя предпочтительнее применяется ***sigmoid()***, иногда ***linear()***.



При использовании данных есть одна проблема. Рассмотрим пример с предсказанием зарплаты по резюме: диапазон значений зарплат y_train от 20 000 до 300 000. Нормализуя данные, мы придаем значению «1» — 300 000, поэтому если будет указана цифра больше 300 000, например, 420 000, то эти данные также преобразуются в «1», и при обратном преобразовании мы получим не 420 000, а 300 000.

Можно, конечно, брать значения min и max с запасом, но тогда выборка ограничивается, и предсказание теряет свою точность.

Нормирование данных

Пример нейронной сети для задачи регрессии

Рассмотрим пример создания нейронной сети на примере оценки зарплаты на базе hh.ru.

Эта база содержит числовые данные, категориальные (пол, график работы и т. д.) и текстовые (опыт работы).

Пол, возраст	ЗП	Ищет работу на должность:	Город	Занятость	График	Опыт (двойное нажатие для полной версии)	Последенее/ нынешнее место работы	Последеняя/ нынешняя должность	Образование и ВУЗ	Обновление резюме	Авто
Мужчина , 29 лет , родился 16 мая 1989	40000 руб.	Специалист по поддержке чата(support team) дом...	Новороссийск , готов к переезду (Анапа, Геленд...	полная занятость	полный день	Опыт работы 3 года 9 месяцев Специалист по по...	ООО "Гольфстрим"	Генеральный директор	Высшее образование 2011 Международный юридиче...	26.04.2019 08:04	Не указано
Мужчина , 38 лет , родился 25 мая 1980	40000 руб.	Системный администратор	Новосибирск , м. Березовая роща , не готов к ...	полная занятость	полный день	Опыт работы 11 лет 11 месяцев Системный админ...	ООО «Завод модульных технологий»	Системный администратор	Высшее образование 2002 Новосибирский государс...	26.04.2019 04:30	Не указано
Мужчина , 35 лет , родился 14 июня 1983	300000 руб.	DevOps TeamLead / DevOps архитектор	Москва , готов к переезду , готов к редким ком...	полная занятость	полный день	Опыт работы 12 лет 11 месяцев DevOps TeamLead...	Банк ВТБ (ПАО)	Начальник отдела методологии разработки (DevOp...	DevOps TeamLead / DevOps архитектор 300 000 ру...	09.04.2019 14:40	Не указано
Мужчина , 33 года , родился 2 августа 1985	180000 руб.	Руководитель IT отдела	Москва , м. Щукинская , не готов к переезду ,...	частичная занятость, полная занятость	удаленная работа, полный день	Опыт работы 15 лет 10 месяцев Руководитель IT...	"Ай-Теко", ведущий российский системный интегр...	Старший системный администратор	Руководитель IT отдела 180 000 руб. Информацио...	09.04.2019 14:39	Имеется собственный автомобиль

Для решения этой задачи наша сеть будет разбита на три нейронки:

1. Первая нейронка будет работать с ***числовыми данными***.
2. Вторая нейронка будет работать с ***простыми текстовыми данными***.
3. Третья нейронка будет работать со ***сложными текстовыми данными***.

Преобразование данных.

1. **Пол** — мужской или женский, будет преобразован в 1 или 0.

   0 — женского пола

   1 — мужского пола
2. **Возраст**. Разбит на 11 классов возрастных промежутков и преобразован в формат one hot encoding(11x).
3. **Город**. Разбит на 4 класса ohe(4x). 0 класс — Москва, 1 класс — Санкт-Петербург, 2 — города-миллионники, 3 — все остальные города. Например, [0, 1, 0, 0], значит, что это Санкт-Петербург.

Нормирование данных

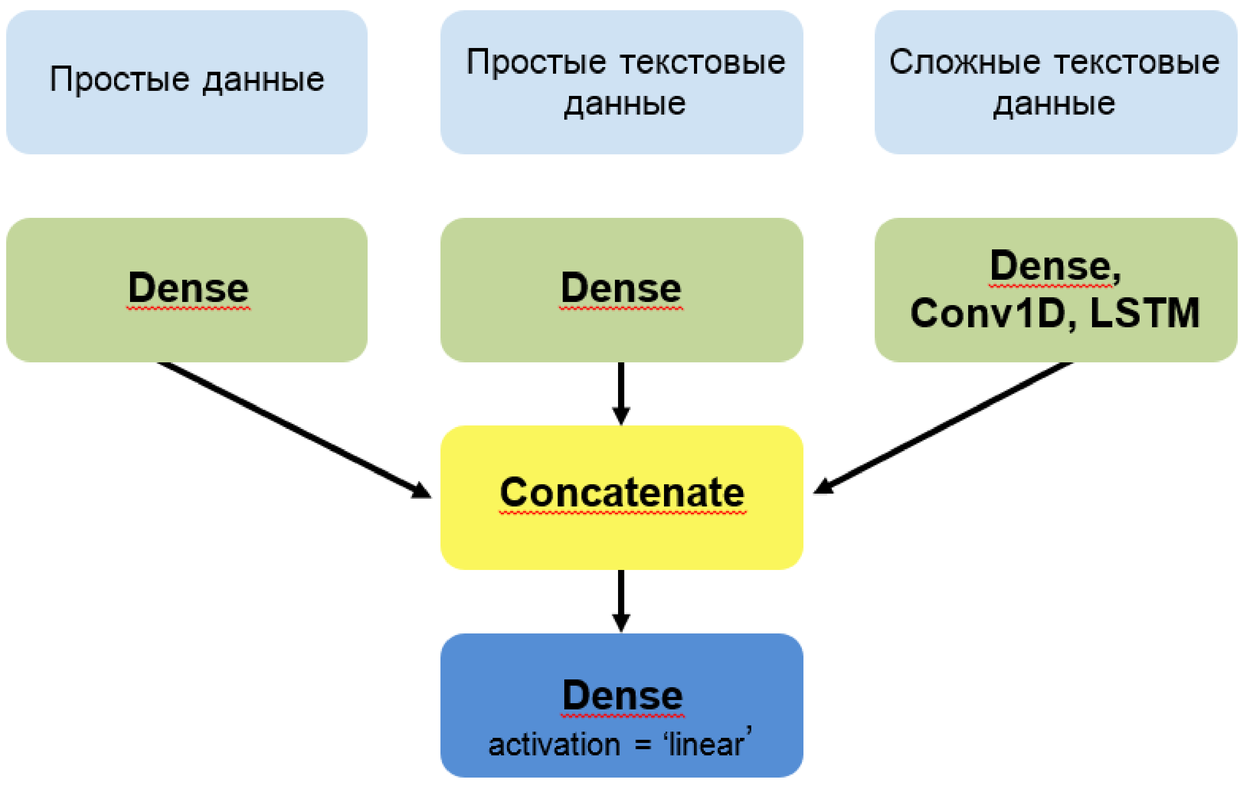
4. **График работы**. 4 класса Multi (то же самое, что и ohe, только может указать несколько категорий одновременно), например, [1, 1, 0, 1].
5. **Занятость**. 4 класса Multi(4x), например, [0, 1, 0, 1].
6. **Образование**. 4 класса Multi(4x), например, [0, 1, 0, 1].
7. **Желаемая должность**. Bag of words. Соберём выборку, соберём в словарь и преобразуем в Bag of words(10).
8. **Прошлая должность**. Bag of words(10).
9. **Опыт работы**. Ohe(11x). Стаж разбит на 11 промежутков (месяцев работы) и преобразован в one hot encoding.
10. **Текстовое описание работы**. Bag of words или Embedding.

Первая нейронка будет работать со следующими данными: пол, возраст, город, график работы, занятость, образование, опыт работы.

Вторая нейронка будет работать с такими данными: желаемая должность, прошлая должность.

Третья нейронка только с данными текстового описания опыта работы.

Схема нашей нейронной сети следующая:



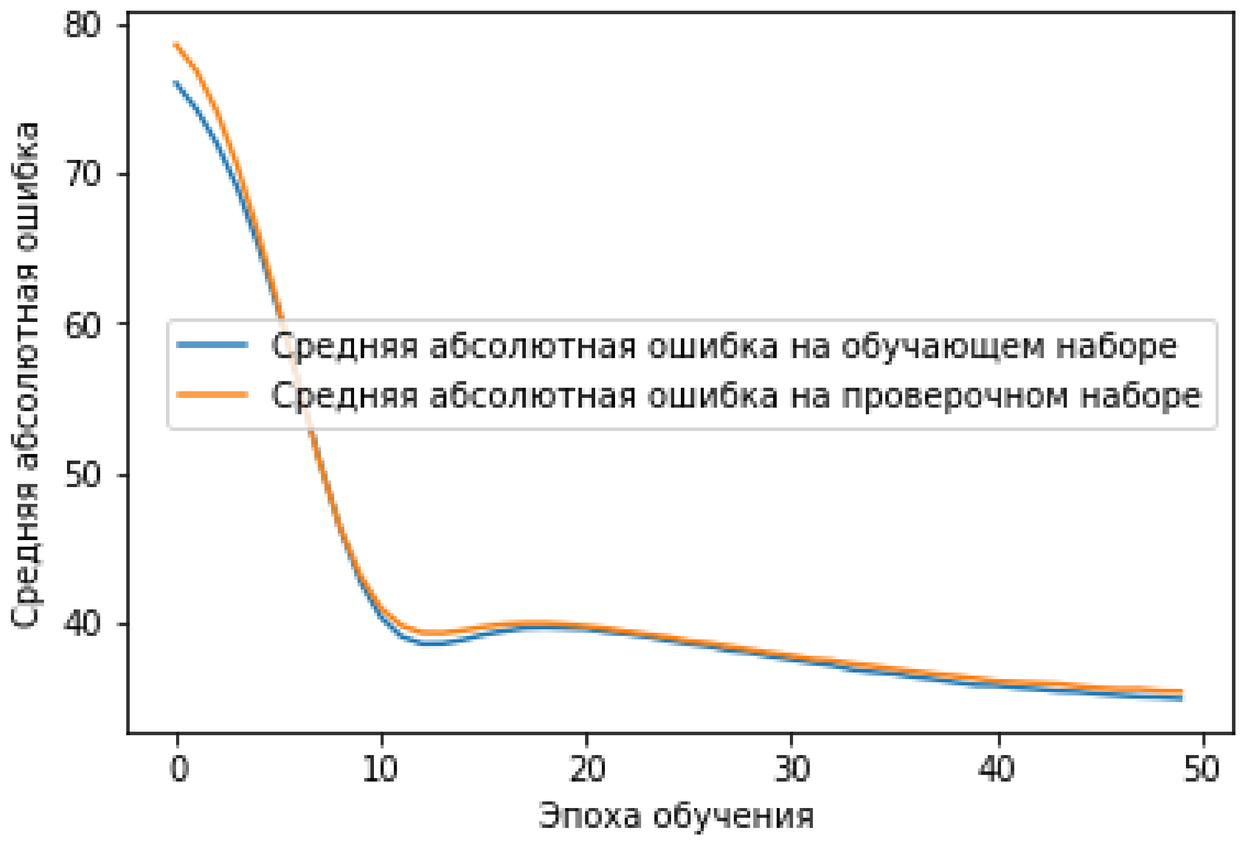
Первая сеть на простых данных имеет следующую архитектуру:

```
# Обучаем модель полученными данными
model = Sequential()
model.add(BatchNormalization(input_shape=(xTrain01.
shape[1],)))
```

Нормирование данных

```
model.add(Dense(100, activation='relu'))
model.add(Dense(1000, activation='tanh'))
model.add(Dense(100, activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(optimizer=Adam(lr=1e-5), loss='mse',
metrics=['mae'])
```

Сеть с 3 Dense слоями с разными функциями активации и выходным слоем с одним нейроном и функцией активации *linear* имеет среднюю абсолютную ошибку на проверочном наборе данных 35.3.



На графике отображена средняя абсолютная ошибка на обучающем и проверочном наборах в зависимости от эпох (всего 50). На графике видно, что если добавить ещё 10-20 эпох, то можно незначительно снизить значение ошибки.

Проверяем сеть на тестовых данных (первые 10).

```
Реальное значение -  40.0  Предсказанное значение -  44.688515  Разница -  4.688514709472656
Реальное значение -  40.0  Предсказанное значение -  72.7303  Разница -  32.73030090332031
Реальное значение -  300.0  Предсказанное значение -  131.92603  Разница -  168.073974609375
Реальное значение -  180.0  Предсказанное значение -  131.33197  Разница -  48.66802978515625
Реальное значение -  40.0  Предсказанное значение -  48.366325  Разница -  8.366325378417969
Реальное значение -  200.0  Предсказанное значение -  123.866234  Разница -  76.1337661743164
Реальное значение -  120.0  Предсказанное значение -  134.75491  Разница -  14.754913330078125
Реальное значение -  50.0  Предсказанное значение -  90.924126  Разница -  40.92412567138672
Реальное значение -  60.0  Предсказанное значение -  95.64962  Разница -  35.64962005615234
Реальное значение -  70.0  Предсказанное значение -  116.3529  Разница -  46.35289764404297
```

Получился довольно неплохой результат.

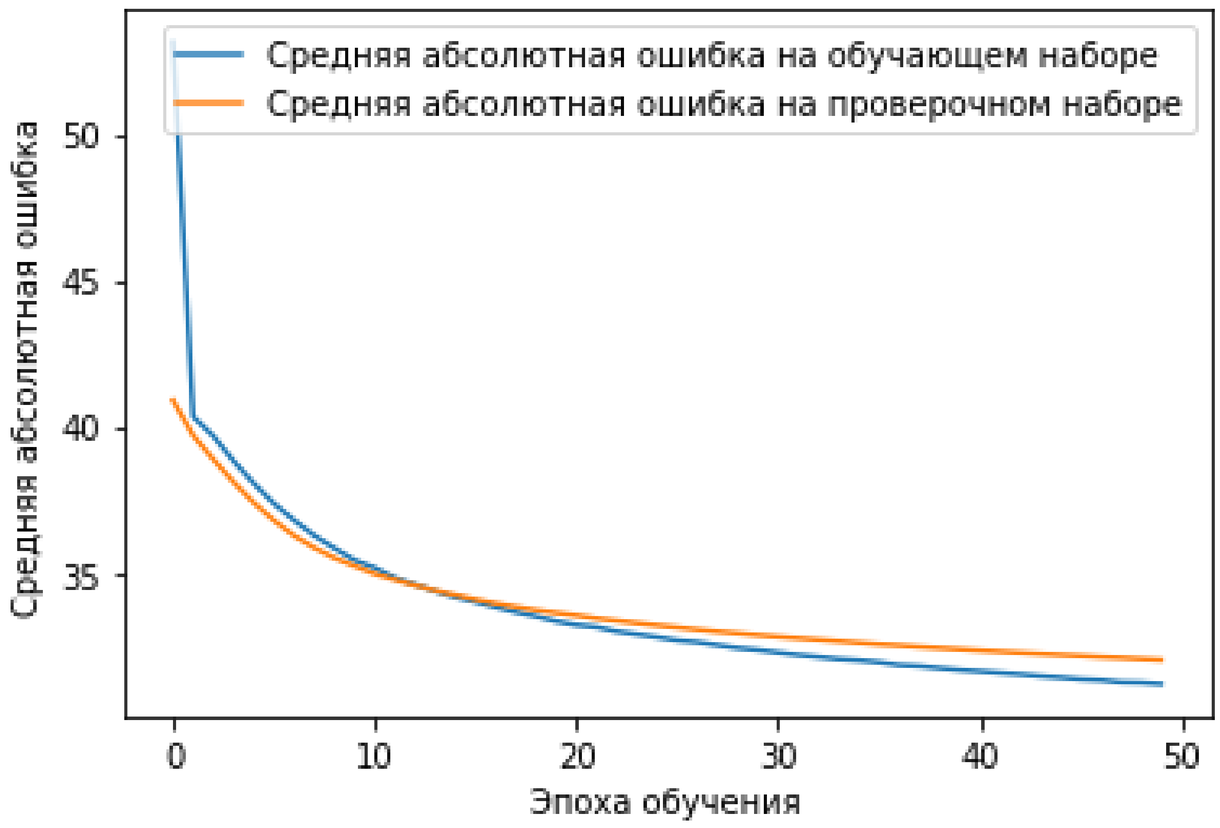
Нормирование данных

Вторая сеть на простых текстовых данных.

Преобразуем текстовые данные в числовые с использованием модели Bag of Words и затем обучаем нейронку на следующей архитектуре:

```
modelTProf = Sequential()
modelTProf.add(Dense(20, activation='relu', input_
dim=(xTrainProf01.shape[1])))
modelTProf.add(Dense(500, activation='relu'))
modelTProf.add(Dense(1, activation='linear'))

modelTProf.compile(optimizer=Adagrad(lr=1e-3), loss='mse',
metrics=['mae'])
```

Значение средней абсолютной ошибки на последней эпохе составляет 32.02.



На графике видно, что если добавить ещё 30 эпох, то можно незначительно снизить значение ошибки.

Третья сеть на сложных текстовых данных.

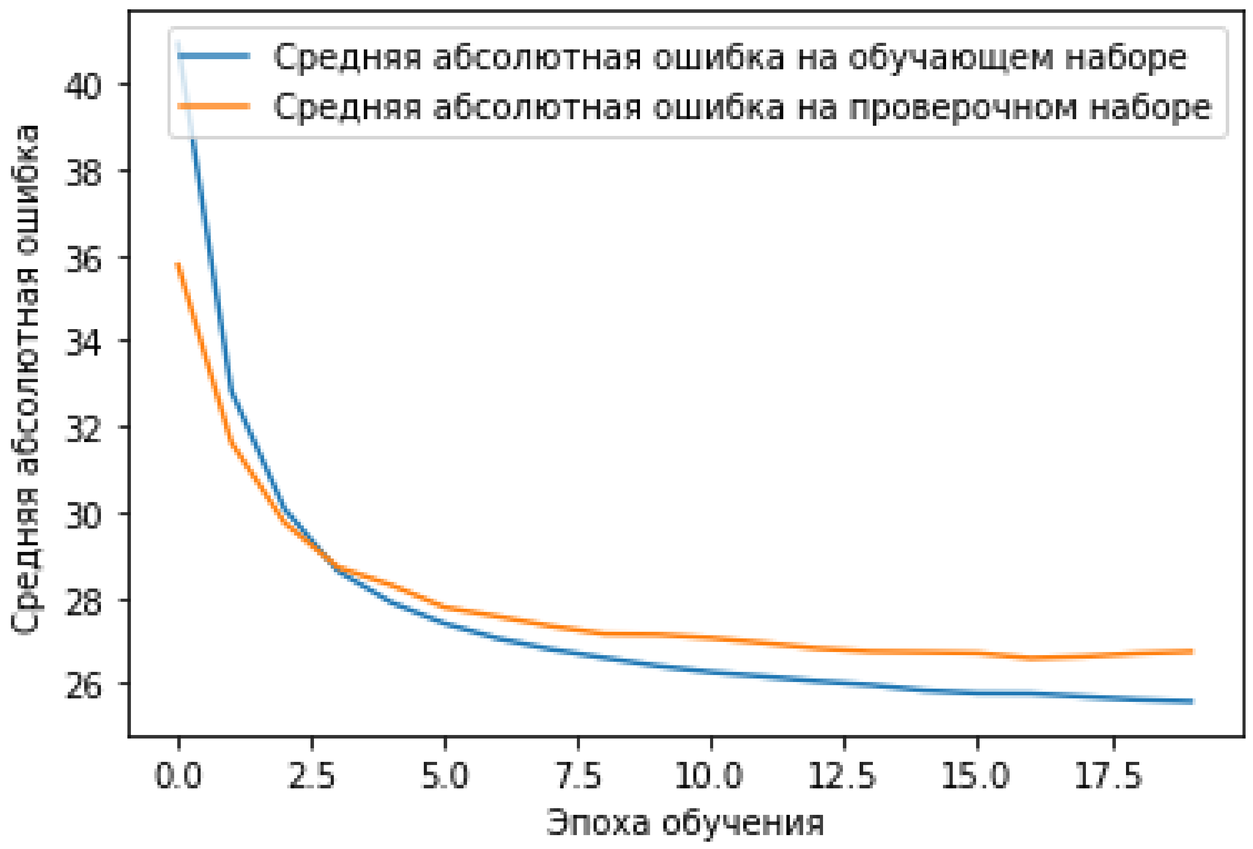
Преобразуем текстовые данные в числовые с использованием модели Bag of Words и затем обучаем нейронку на следующей архитектуре:

```
modelTRez = Sequential()
modelTRez.add(Dense(20, activation='relu',input_
dim=(xTrainRez01.shape[1])))
modelTRez.add(Dense(500, activation='relu'))
modelTRez.add(Dropout(0.3))
modelTRez.add(Dense(1, activation='linear'))
```

Нормирование данных

```
modelTRez.compile(optimizer=Adam(lr=1e-3), loss='mse',
metrics=['mae'])
```

Обучаем на 20 эпохах. Значение средней абсолютной ошибки на последней эпохе составляет 26.73.



Далее делаем составную нейронку, которая состоит из наших трех предыдущих.

```
input1 = Input((xTrain01.shape[1],))      # Входной слой
для первой нейронки
input2 = Input((xTrainProf01.shape[1],)) # Входной слой
для второй нейронки
input3 = Input((xTrainRez01.shape[1],))  # Входной слой
для третьей нейронки
x1 = BatchNormalization()(input1) # Создаем ветку x1
x1 = Dropout(0.5)(x1)
x1 = Dense(10, activation="relu")(x1)
x1 = Dense(1000, activation="relu")(x1)
x1 = Dense(100, activation = "relu")(x1)

x2 = BatchNormalization()(input2) # Создаем ветку x2
x2 = Dense(25, activation="relu")(input2)
x2 = Dense(8, activation="tanh")(x2)
x2 = Dense(5, activation = "elu")(x2)
```

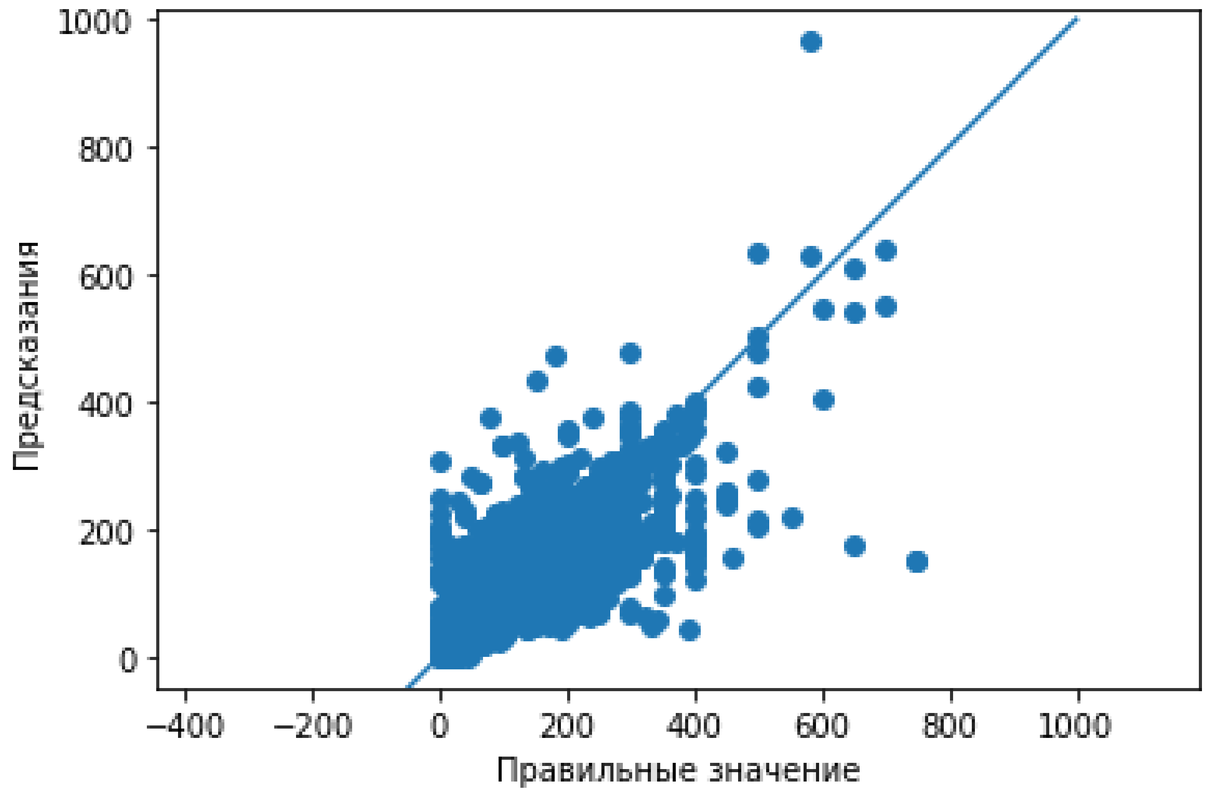
Нормирование данных

```
x3 = BatchNormalization()(input3) # Создаем ветку x3
x3 = Dense(1000, activation=”tanh”)(input3)
x3 = Dense(20, activation = “elu”)(x3)
x3 = Dense(5, activation = “elu”)(x3)

x = concatenate([x1, x2, x3]) # Объединяем все три ветки

x = Dense(15, activation=’linear’)(x)
x = Dropout(0.5)(x)
x = Dense(1, activation=’relu’)(x) # Финальный нейрон
делает регрессию

model = Model((input1, input2, input3), x) # В Model
загружаем стартовые и последнюю точки

model.compile(optimizer=Adam(lr=1e-3), loss=’mse’,
metrics=[‘mae’])
```

Обучаем на 150 эпохах. Значение средней абсолютной ошибки на последней эпохе составляет 17.69.

Нормирование данных

Выведем график зависимостей зарплаты от предсказанной зарплаты.



Если точки находятся на прямой линии графика (диагональ), значит, предсказание абсолютно верное. Разброс небольшой. Есть несколько сильных отклонений, их можно исправить, если почистить базу.