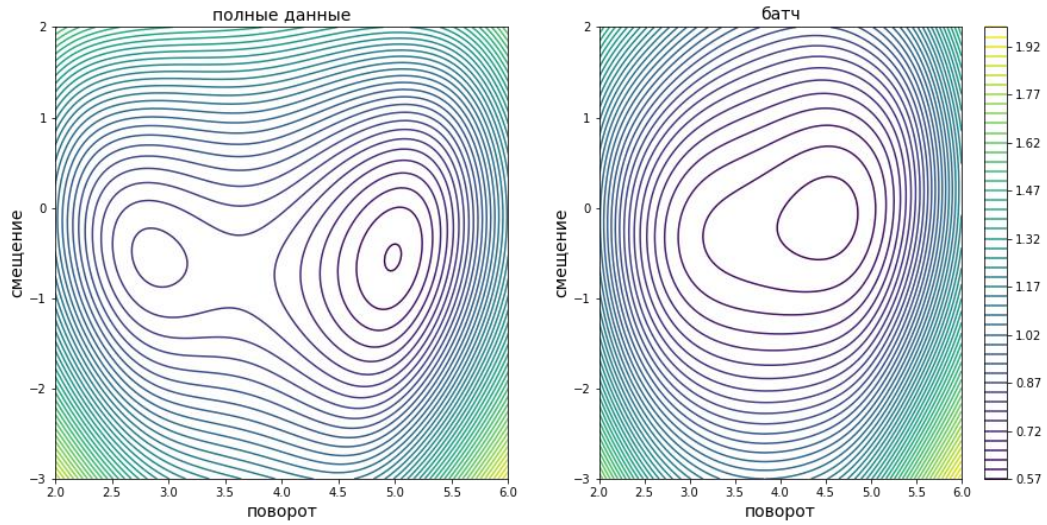

Рисунок 3.1 – Целевая функция по параметрам

Примечание – Источник: собственная разработка.

Теперь, случайным образом, выделим из общего набора данных батч. Он представлен на рисунке К.2 или, еще ценнее, для нас рисунок 3.1 справа. По этим рисункам можно увидеть, что второе решение попросту выродилось при использовании батча. Можно, конечно, заметить, что могли просто выпасть так наблюдения в батч, но в реальных условиях в алгоритме за одну эпоху отрабатывает множество батчей и, похоже, можно доказать, что чаще всего в них будет сохраняться наиболее заметная статистическая закономерность, что приведет к тому что алгоритм будет сходиться к более заметным минимумам.

Вернемся к загрузчику – реализации обучения по эпохам в pytorch. Загрузчик данных должен представлять собой экземпляр класса «torch.utils.data.DataLoader». Для его создания, в конструктор надо передать экземпляр-наследник класса «torch.utils.data.DataSet», с переопределенными методами «__len__» и «__getitem__». Первый, из названных методов, должен возвращать длину используемого набора данных, второй – пердоставлять доступ к элементам по переданному индексу. Таким образом в «pytorch» достигают независимости от формата входных данных, разработчику модели предоставлена возможность описать как ведет себя его набор данных. Наша реализация наследника «torch.utils.data.DataSet» представлена на листинге К.3.

Не синтаксически, но фактически обязательно, знать про именованный аргумент «batch_size» загрузчика данных. Названный аргумент позволяет регулировать размер батча. Размер батча, пожалуй, самый важный параметр касающийся данных на этапе обучения сети – малый батч позволит обойти локальные минимумы, вместе с тем, целевая функция теряет точность, потому и оценки параметров на больших батчах будут точнее.

Возвращаясь к алгоритму обучения сети, в методе «fit» мы запускаем цикл по загрузчику тренировочных данных. Этот цикл обеспечивает обход батчей, а внутри уже самый обычный алгоритм обратного распространения ошибки.