1 Оптимизация систем с помощью экспериментов

В ЭТОЙ ГЛАВЕ

- ✓ Оптимизация инженерных систем
- ✓ Понятие эксперимента
- ✓ Уникальная ценность эксперимента

В последние 20 лет значительно возрос интерес к разработке экспериментальных методов для оценки и совершенствования таких инженерных систем, как веб-приложения, торговые роботы и программные среды. Повысилась эффективность и степень автоматизации экспериментальных методов. Сфера их применения распространилась на такие крупные системы, как поисковые машины и социальные сети. В результате улучшение производительности работающих производственных систем в настоящее время может происходить непрерывно и автоматически.

С помощью экспериментов инженеры оценивают влияние изменений в системе и определяют оптимальный набор значений ее параметров. Будем называть это *экспериментальной оптимизацией*.

В этой книге показаны различные экспериментальные методы, которые используют профессионалы на торговых площадках и в информационных технологиях.

Речь пойдет о системах, разработанных специалистами трех различных направлений:

- инженерами по машинному обучению (ИМО);
- количественными трейдерами («кванттрейдерами», или «квантами»);
- программистами.

Инженеры по машинному обучению разрабатывают такие веб-приложения, как поисковые машины, рекомендательные системы и контекстная реклама. Кванттрейдеры создают торговых роботов, а программисты — инфраструктуру и инструменты, например веб-серверы, компиляторы и системы обработки событий.

Процесс совершенствования систем во всех этих отраслях представляет собой бесконечный цикл, включающий одни и те же этапы (рис. 1.1).

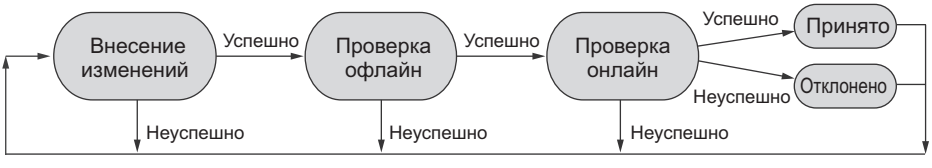

Рис. 1.1. Этапы рабочего процесса управления изменениями. (1) Сначала новая идея выполняется в виде изменений в коде. (2) Затем, как правило, проводится некоторое офлайн-тестирование, чтобы отбросить идеи, которые могут негативно повлиять на бизнес-метрики. (3) Изменения включаются в рабочее ПО, и бизнес-метрики измеряются онлайн. Если изменения успешно проходят проверку, то они остаются в системе. Весь процесс повторяется с начала. Так в систему непрерывно добавляются надежные изменения, улучшающие производительность

Таким образом разрабатываемые системы постепенно совершенствуются. Один человек или команда предлагают идеи, которые, по их мнению, улучшат систему, затем каждая идея проходит все этапы. Хорошие идеи остаются в системе, плохие — отклоняются:

1. *Внесение изменений*. Сначала разработчик выполняет идею — изменяет код и обновляет программное обеспечение системы. На этом этапе код проходит стандартные процедуры контроля качества, такие как код-ревью и юнит-тестирование. Изменения кода, которые успешно прошли все тесты, переходят на следующий этап.
2. *Проверка офлайн* — влияние изменений кода на бизнес-метрики проверяется офлайн, вне рабочей среды. Для приблизительной оценки бизнес-метрик, таких как доход или ожидаемое количество кликов на объявлении, в этой проверке обычно используют данные, полученные в ходе предыдущей эксплуатации системы. Если оценка покажет, что изменения в коде ухудшают

бизнес-метрики, то изменения отклоняются. В противном случае изменения переходят на заключительный этап.

3. *Проверка онлайн* — изменения в коде выгружаются в продакшен-версию системы, где оценивается их влияние на бизнес-метрики. Изменения кода могут потребовать некоторой настройки системы — установки флагов или численных параметров. В таком случае инженеры оценивают бизнес-метрики для разных конфигураций, чтобы выявить самую удачную. Изменения кода, которые не улучшают бизнес-метрики, отклоняются. В противном случае изменения остаются в системе. В результате достигается улучшение системы.

В этой книге рассматривается последний этап — «проверка онлайн». На этом этапе вы применяете экспериментальные методы на работающей системе. Такие эксперименты ценны тем, что приносят результаты измерений из реальной системы, а эту информацию нельзя получить никаким другим способом. Но эксперименты требуют времени. Некоторые из них на протяжении дней или недель. И не без риска. Если вы запускаете эксперимент, то можете потерять деньги, оттолкнуть пользователей либо заработать плохую репутацию или негативные отзывы, если пользователи заметят ваши действия и начнут жаловаться. Поэтому вам нужно проводить измерения максимально быстро и аккуратно, чтобы уменьшить негативные последствия от нерабочих идей — назовем их для краткости *затратами* — и извлечь наибольшую выгоду от идей рабочих.

Чтобы извлечь максимальную пользу из нового фрагмента кода, вам необходимо оптимально настроить его. Это похоже на настройку радио или гитарной струны. Вы просто крутите ручку влево или вправо и слушаете, что получается. Если повернуть ручку слишком сильно, то радио начнет шуметь, а гитара зазвучит резко или глухо. То же самое с параметрами конфигурации кода (их часто сравнивают с такими ручками (*knobs*), которые я только что упомянул). Необходимо выбрать такие значения параметров, при которых внесенные изменения окажут наиболее положительное влияние на бизнес-метрики — доход, количество кликов и пр. Проведение экспериментальной оптимизации — это дорогостоящая вещь, и это выделяет ее среди прочих методов оптимизации.

В этой главе рассматриваются экспериментальные методы оптимизации систем в перечисленных выше областях — машинном обучении, кванттрейдинге и программировании. Мы увидим, с какими системами работают специалисты в этих областях, какие метрики измеряют и как у них организован каждый этап.

Возможно, вам доводилось слышать и о других способах оценки изменений в системе. В первую очередь это оценка на основе модели, использование знаний предметной области и имитационное моделирование. Мы обсудим, почему эти необходимые инструменты не могут заменить измерения, сделанные во время экспериментов.

1.1. ПРИМЕРЫ РАБОЧИХ ПРОЦЕССОВ

Специалисты, о которых шла речь выше, работают в различных предметных областях, но рабочие процессы управления изменениями у них похожи. Каждый процесс можно рассматривать как частный случай общего рабочего процесса, показанного на рис. 1.1: внесение изменений, проверка офлайн, проверка онлайн. Давайте детально рассмотрим примеры рабочих процессов инженера по машинному обучению, кванттрейдера и программиста.

1.1.1. Рабочий процесс в системе машинного обучения

Представьте ИМО, которые работают над новостным веб-сайтом. Процесс внесения изменений в их случае может выглядеть как на рис. 1.2.

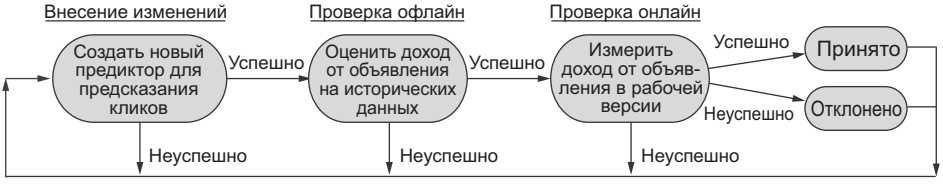

Рис. 1.2. Пример рабочего процесса управления изменениями в системе МО на примере новостного веб-сайта. Сайт содержит компонент МО, который предсказывает клики на новостных статьях (предиктор). (1) ИМО добавляет новый предиктор. (2) Основываясь на логировании кликов пользователей и конверсии рекламного объявления, делается оценка дохода от объявления с новым предиктором. (3) Новый предиктор включают в работу и снова измеряют доход от объявления. Если предиктор улучшил доход, то его оставляют в системе

Главный компонент машинного обучения — предиктор, который предсказывает, на каких статьях в новостях кликнет пользователь. Предиктор может принимать на вход различные параметры, например демографические данные, предыдущую активность пользователя на этом веб-сайте, заголовки и содержание новостных статей. На выходе предиктор выдает оценку вероятности того, что конкретный пользователь кликнет на данной статье. Веб-сайт может использовать эти предсказания для ранжирования и сортировки новостей, чтобы расположить самые интересные новости выше остальных на главной странице.

На рис. 1.2 изображен процесс управления изменениями в коде новостного веб-сайта. Когда ИМО посещает идея улучшить предиктор — добавить новую функцию или выбрать новую модель, — эта идея проходит следующие стадии:

1. *Внесение изменений* — ИМО добавляет предиктор и проверяет его на имеющихся данных. Если на этих данных новый предиктор выдает предсказания лучше, чем старый, то новый предиктор переходит на следующий этап.

2. *Проверка офлайн* — целью бизнеса является не просто улучшение предсказаний количества кликов, а увеличение дохода от рекламных объявлений, размещенных на этом веб-сайте. Перевести улучшение предсказаний в увеличение дохода непросто, но методы, которые дают полезные оценки для некоторых систем, все-таки существуют. Если оценки выглядят неплохо, то предиктор переходит на следующий этап.
3. *Проверка онлайн* — ИМО включает предиктор в рабочую версию программы, и реальные пользователи видят заголовки, ранжированные с помощью этого предиктора. ИМО измеряет доход от рекламного объявления и сравнивает его с доходом, который система показывала при старом предикторе. Если новый предиктор улучшает доход от объявления, то он остается в системе.

Помимо предиктора для кликов, у новостного веб-сайта может быть много других компонентов. Каждый компонент проходит такой же цикл, чтобы инженеры могли убедиться, что доход от объявления стабильно улучшается.

ИМО работают над различными видами систем. Сортировка новостных заголовков по частоте кликов — только один пример широкого класса систем, называемых *рекомендательными системами*. Такие системы используются для ранжирования видео, музыки, постов в социальных сетях, потребительских товаров и многого другого. Поисковые системы похожи на системы машинного обучения, потому что они могут ранжировать результаты поиска для конкретного пользователя. Еще один вид систем машинного обучения — таргетированная реклама, когда объявления выбираются для показа на основе предпочтений пользователя. Теперь давайте обратимся к финансам и посмотрим на рабочий процесс квантрейдеров.

1.1.2. Рабочий процесс в количественном трейдинге

Этот процесс в квантрейдинге очень похож на рабочий процесс в машинном обучении. Отличие состоит в деталях, например в предсказаниях, которые нужно сделать (рис. 1.3).

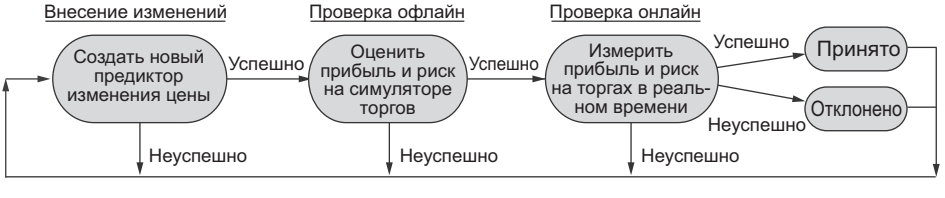

Рис. 1.3. Пример рабочего процесса управления изменениями в квантрейдинге на примере изменения стратегии торгового робота. Стратегия содержит предиктор изменения цены. (1) Квант создает новый предиктор. (2) Запускает симулятор торгов и оценивает прибыль и риск на данных прошлых периодов. (3) Квант измеряет актуальную прибыль и риск в процессе торговли в реальном времени. Если новый предиктор увеличивает прибыль и/или уменьшает риск, то он остается в системе

Квант разрабатывает стратегию торгового робота — части его программного обеспечения, которая выдает ордера на покупку/продажу на бирже в надежде, как говорят кванты, купить подешевле и продать подороже. Ключевой компонент — модель, которая предсказывает изменения цены финансового инструмента, выставленного на торги (например, акций). Если предсказано увеличение цены, то это удачное время для подачи ордера на покупку. Аналогично, если предсказано снижение цены, то это удачное время для подачи ордера на продажу. Бизнес-метрикой для этой системы является прибыль. И еще риск. Кванты хотят увеличить прибыль и уменьшить риск. Часто (наверное, даже обычно) при оптимизации системы во внимание принимают не одну, а несколько бизнес-метрик. В главе 7, раздел 3, мы подробно рассмотрим этот важный практический момент.

На рис. 1.3 изображен рабочий процесс в кванттрейдинге. Изменения торговой стратегии проходят через следующие этапы:

1. *Внесение изменений* — квант создает новый предиктор изменения цены и проверяет на исторических данных рынка, что новый предиктор выдает более точные предсказания, чем старый.
2. *Проверка офлайн* — предикторы для предсказания лучшей цены не гарантируют более высокую прибыль (или более низкий риск). Поэтому необходимо проверить всю торговую стратегию — предиктор, ордера на покупку/продажу и т. д. — на симуляторе торгов с использованием исторических данных рынка (это также называется бэктестом). Симулятор торгов генерирует предсказания и моделирует покупки/продажи, чтобы оценить прибыль и риск. Если стратегия улучшилась, то предиктор переходит на следующий этап.
3. *Проверка онлайн* — предиктор выводят на реальные торги, где размещены настоящие заявки, а деньги и акции переходят из рук в руки. Только реальные торги могут показать актуальную прибыль и риск торговой стратегии. Если предиктор уменьшает прибыль или увеличивает риск, то изменения в нем отменяются.

Кванты обычно работают с одним из двух типов торговых стратегий: принципал-трейдинг или агентский трейдинг. Принципал-трейдинг — это торговля непосредственно для прибыли оператора (кванта или компании, в которой работает квант). Агентский трейдинг оказывает услуги торговли на бирже и торгует от имени клиентов, помогая им снизить расходы.

Существует множество вариаций этих стратегий. Торговать можно акциями, фьючерсными контрактами, опционами и многими другими финансовыми продуктами. Каждый тип продукта, как правило, имеет в мире несколько торговых бирж.

Еще один ключевой компонент стратегии — это ее время действия. Принципал-трейдер некоторое время владеет акцией (или другим инструментом), прежде

чем продать ее. Это могут быть минуты, часы, дни или недели. Иногда даже целые месяцы или, наоборот, секунды. Разные временные рамки требуют разных предикторов и разных подходов к оценке рисков.

Жизненные циклы изменений в машинном обучении и квантрейдинге похожи, потому что похожи и сами системы. Они обычно состоят из прогнозной модели, основанной на данных, и некоторого кода, который решает, как использовать предиктор. Процесс у программистов немного отличается, и это наша следующая тема.

1.1.3. Рабочий процесс в разработке ПО

Программисты работают над широким спектром систем, но задачи программистов не связаны с построением моделей на основе данных (это отличает их от ИМО и квантов). Они разрабатывают компиляторы, системы кэширования, веб-серверы, инфраструктуру торговых систем (на которых запускаются торговые стратегии) и многое другое.

Для примера давайте возьмем следующую задачу: необходимо улучшить время отклика поисковой системы и снизить показатель отказов, то есть вероятность того, что пользователь уйдет с веб-сайта, если страница будет загружаться слишком долго. На рис. 1.4 изображен процесс управления изменениями у программистов.

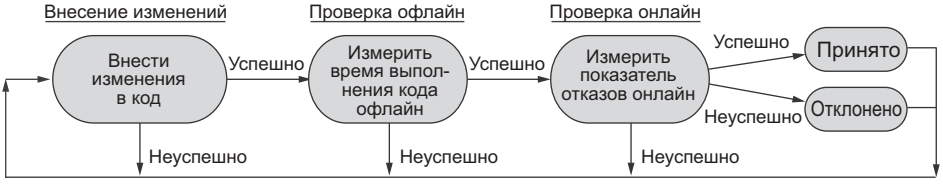

Рис. 1.4. Пример процесса управления изменениями в разработке ПО серверной части поисковой системы. Прежде чем отправить ответ пользователю, сервер запрашивает, собирает и преобразовывает необходимые данные. (1) Программист изменяет код, отвечающий за преобразование данных. (2) Затем программист измеряет время работы нового кода офлайн и проверяет, что новый код преобразовывает тестовые данные быстрее старого. (3) Когда выходит рабочая версия кода, программист измеряет показатель отказов — метрику, важную для бизнеса, и проверяет, как новый код повлиял на эту скорость. Если новый код снизил показатель отказов, то изменения остаются в системе

Представим, что программисты разрабатывают поисковую систему — веб-сервер, который отвечает на запросы пользователей: запрашивает данные из внутренних источников, преобразовывает эти данные и отправляет пользователю ответ в заданном формате. Для пользователей крайне важно получать ответ от веб-сервера

быстро. Если ответ будет идти слишком долго, то пользователь может уйти со страницы, так и не дождавшись ответа.

Существует много причин, способных замедлить скорость ответа веб-сервера (медленный браузер, медленный интернет, отсутствие необходимых данных в кэше и т. д.), но программисты выдвигают гипотезу, что замедление происходит именно на этапе преобразования данных. Чтобы устранить проблему, гипотезу необходимо провести через следующие этапы:

1. *Внесение изменений* — программисты вносят в код изменения, которые, по их мнению, должны ускорить этап преобразования данных.
2. *Проверка офлайн* — код запускается офлайн на разных наборах данных, которые составлены на основе предыдущих запросов пользователей, измеряется время выполнения кода. Если время отклика поисковой системы уменьшается, то код переходит на следующий этап.
3. *Проверка онлайн* — изменения в коде включаются в продакшен-версию, где поисковая система отвечает на запросы реальных пользователей. Программисты измеряют показатель отказов и сравнивают его с показателем отказов до внесения изменений. Если новый код снижает показатель отказов, то изменения остаются в системе.

Обычно инженеры генерируют много нестандартных идей, чтобы улучшить системы, над которыми они работают. Представим, что новые идеи — это сырье, тогда процесс управления изменениями — это фабрика, которая уверенно и надежно перерабатывает сырье в улучшения системы.

Каждый цикл рабочего процесса заканчивается измерением бизнес-метрик онлайн. Измерения производятся с помощью экспериментов на работающей системе.

1.2. ИЗМЕРЕНИЕ С ПОМОЩЬЮ ЭКСПЕРИМЕНТОВ

В торговле на бирже и в информационных технологиях используются сложные системы. Эта сложность затрудняет оценку влияния изменений на бизнес-метрики. Представьте веб-сайт, который продает какой-то товар. Для этого веб-сайта важна такая бизнес-метрика, как ежедневный доход, — денежная сумма, которую пользователи платят компании каждый день. Эта сумма зависит от качества товара, его конкурентоспособности, от того, сколько людей знают о товаре, сколько людей его уже купили, насколько люди склонны совершать покупки в тот или иной день (например, в выходные или в «Черную пятницу»), насколько понятен интерфейс веб-сайта, насколько легко им пользоваться и т. д. На ежедневный доход влияет очень много факторов, и большинство из них компания не может контролировать.

Если вы внесли изменения в такую систему и измерили ежедневный доход, то как вы узнаете, что именно на него повлияло — ваше изменение или какой-то сторонний фактор? Стал бы ежедневный доход больше или меньше, если бы вы не внесли изменения? Но главное, что с ним произойдет в будущем, если вы оставите или уберете свои изменения? На эти вопросы можно ответить с помощью экспериментов.

1.2.1. Экспериментальные методы

Эксперименты учитывают эффект только от внесенных в систему изменений, игнорируя остальные факторы, которые могут повлиять на бизнес-метрику. Удивительно, но эксперименты даже учитывают влияние факторов, которые неизвестны вам как инженеру (мы подробно рассмотрим это в главе 2). Эксперимент способен отделить влияние ваших изменений от всех остальных факторов, и именно поэтому он является хорошим инструментом для оценки влияния изменений на бизнес.

Эксперименты безусловно полезны, но за них приходится расплачиваться. Для запуска экспериментов требуется время, они могут снизить производительность системы (например, если внесенные изменения не улучшают, а ухудшают ее работу) или повредить ее (например, из-за ошибки в новом коде). Давайте попробуем разобраться, как снизить эти риски, чтобы получить наибольшую выгоду от экспериментов. В главе 2 мы рассмотрим, как спланировать эксперимент, чтобы свести к минимуму время его проведения, но при этом получить нужные результаты. Все последующие главы о методах проведения экспериментов, с главы 3 по главу 6, рассматривают способы снижения рисков в различных ситуациях. В главах 3 и 5, посвященных алгоритмам многорукого бандита, рассматривается, как создать адаптивный план эксперимента, то есть план, который совершенствуется в процессе эксперимента, пока идет накопление результатов измерений.

Напомню, что некоторые изменения в системе требуют оценки бизнес-метрик на различных конфигурациях системы и выбора наиболее удачной конфигурации. Это значительно увеличивает стоимость проведения измерений. Главы 4 и 6 рассматривают метод анализа поверхности отклика и метод байесовской оптимизации соответственно, которые с помощью статистического анализа делают предположение о том, какая конфигурация системы наиболее эффективна, и уменьшают итоговое количество измерений.

Перечисленные методы широко использовались в различных отраслях индустрии. Некоторые из них применяются уже 10 лет, а какие-то даже 70 лет. Эти методы также популярны в областях, в которых работаю я, — количественный трейдинг и социальные сети. В этих областях есть все условия для проведения экспериментов, потому что их системы активно взаимодействуют с миром.

Торговые системы отправляют тысячи или даже десятки тысяч заявок в день. Веб-сайты обрабатывают от тысячи до миллиарда (для самых крупных веб-сайтов) запросов в день. Каждое взаимодействие открывает возможности для эксперимента.

Основываясь на личном опыте, разговорах с коллегами и интервью, проведенных для написания этой книги, я постарался ограничить материал и оставить только те методы, которые хорошо показали себя на практике. В этой книге я собрал описания методов, примеры из жизни, практические задачи и подводные камни.

1.2.2. Практические задачи и подводные камни

При применении экспериментальных методов предполагается, что бизнес-метрики известны. В главе 7 говорится о том, как выбрать бизнес-метрики, и о том, что лучше учитывать сразу несколько бизнес-метрик. Также в главе 7 рассказано о том, как интерпретировать результаты экспериментов и как эта интерпретация может усложниться, когда в эксперименте участвуют несколько бизнес-метрик или заинтересованных лиц.

Наконец, глава 8 рассматривает, как реальные данные могут отклоняться от первоначальных предположений, сделанных при разработке эксперимента, а также общие источники ошибок в интерпретации результатов.

Прежде чем углубляться в детали экспериментальных методов, вам стоит подумать: нужно ли вам вообще экспериментировать? Это займет время и силы на подготовку инструментов, необходимых для разработки эксперимента, оценки изменений в системе, а также анализа этой оценки. Чтобы оправдать эти усилия, вы должны получить что-то взамен. В следующем разделе рассмотрены некоторые «за» и «против» экспериментов.

1.3. ПОЧЕМУ ЭКСПЕРИМЕНТЫ НЕОБХОДИМЫ

Наверняка каждый программист знаком с предостережением, которое приписывают Дональду Кнуту, о том, что «преждевременная оптимизация — корень всех зол» — то есть вместо того, чтобы с самого начала внедрять идеи, которые, как вы ожидаете, заставят ваш код работать быстрее (или лучше), сначала напишите простой код для решения проблемы, придумайте способ измерить время выполнения этого кода, а затем проверяйте свои идеи по одной, чтобы увидеть, какие из них действительно ускоряют работу кода. Слишком сложно рассуждать сразу обо всех факторах, которые могут повлиять на скорость, — это может быть вся кодовая база системы, архитектура компьютера, операционная система и т. д., поэтому необходимо использовать тесты.