

профориентация
сесреты
выбор профессии

Самиздат

Архитекторы
Синтеза
Инженерия
интеллекта от идеи
до продакшена

Турист Мартовский

Турист Мартовский Архитекторы Синтеза Инженерия интеллекта от идеи до продакшена

<https://litres.ru/74609738>

SelfPub; 2026

Аннотация

Практическое руководство по созданию, обучению и масштабированию систем машинного обучения нового поколения.

В эпоху, когда искусственный интеллект перестал быть научной фантастикой и превратился в рабочий инструмент, роль AI/ML инженера стала одной из ключевых в технологическом мире. Но между написанным в Jupyter Notebook прототипом и отказоустойчивой системой, обслуживающей миллионы пользователей, лежит пропасть.

Книга «Архитекторы Синтеза» — это не просто сборник алгоритмов. Это настольная библия для инженера, который хочет не просто "поиграть с нейросетями", а строить промышленные решения. Книга проведет вас по полному циклу жизни ML-системы: от постановки бизнес-задачи и сбора данных до

развертывания в Kubernetes и мониторинга деградации моделей (Data Drift).

Вы научитесь мыслить как архитектор: балансировать между точностью модели, скоростью инференса и стоимостью вычислений. Это руководство объединяет теорию машинного обучения, программную инженерию и MLOps в единую дисциплину

Турист Мартовский Архитекторы Синтеза Инженерия интеллекта от идеи до продакшена

Введение

Эта книга написана для тех, кто хочет не просто обучить модель в Jupyter Notebook, а построить систему, которая будет работать в реальном мире. Для тех, кто понимает, что AI/ML Engineer — это не просто человек, который знает PyTorch, а архитектор, который соединяет данные, алгоритмы, инфраструктуру и бизнес в единое целое.

Я написал эту книгу, потому что видел слишком много талантливых специалистов, которые умеют строить модели, но не умеют их развертывать. Видел проекты, которые умирали на стадии прототипа, потому что никто не подумал о том, как модель будет работать под нагрузкой. Видел компании, которые тратили миллионы на GPU, но не могли объяснить, почему их модель ошибается.

Эта книга — попытка закрыть пропасть между академическим знанием и промышленной практикой. Она проведет

вас через все этапы жизни ML-системы: от понимания бизнес-задачи до мониторинга деградации модели в продакшене.

Часть I: Фундамент. Инженерное мышление в мире данных

Глава 1. Эволюция роли: От аналитика до архитектора систем

1.1. Кто такой AI/ML Engineer сегодня

В 2012 году, когда AlexNet выиграл ImageNet, никто не говорил о профессии "ML Engineer". Были исследователи, которые писали статьи, и разработчики, которые писали код. Но между ними образовалась пропасть: исследователи не умели строить продакшен-системы, а разработчики не понимали математику за моделями.

Сегодня AI/ML Engineer — это гибридная роль. Он понимает:

Математику: как работают градиентные методы, почему трансформеры эффективны, что такое переобучение.

Программную инженерию: как писать чистый, тестируемый код, как проектировать API, как работать с Git.

Инфраструктуру: как развернуть модель в Kubernetes, как

настроить CI/CD, как мониторить систему.

Бизнес: как перевести задачу "увеличить конверсию" в метрику ROC-AUC, как объяснить stakeholder'ам, почему модель ошибается.

Разница между Data Scientist и ML Engineer:

Аспект Data Scientist ML Engineer

Фокус Исследование, анализ Создание систем

Код Jupyter Notebook Production-ready код

Метрики Accuracy, F1 Latency, Throughput, Cost

Инструменты Pandas, Scikit-learn PyTorch, Docker, Kubernetes

Результат Отчет, презентация Работающий сервис

1.2. Жизненный цикл ML-проекта (MLLC)

ML-проект — это не линейный процесс. Это цикл, который повторяется снова и снова.

Этап 1: Постановка задачи

Вы получаете запрос от бизнеса: "Хотим, чтобы система предсказывала отток клиентов". Вы должны перевести это на язык ML: "Задача бинарной классификации. Целевая метрика — ROC-AUC > 0.85. Ограничение — инференс < 100 мс".

Этап 2: Сбор и подготовка данных

80% времени. Вы обнаруживаете, что данные хранятся в 5 разных базах, что 30% значений пропущены, что целевая переменная размечена с ошибками.

Этап 3: Исследование и прототипирование

Вы пробуете разные модели, делаете feature engineering, настраиваете гиперпараметры. Это этап Jupyter Notebook.

Этап 4: Инженерия и продакшен

Вы превращаете прототип в сервис. Пишете API, настраиваете CI/CD, разворачиваете в контейнерах.

Этап 5: Мониторинг и поддержка

Вы следите за метриками, обнаруживаете Data Drift, переобучаете модель.

1.3. Постановка задач: от бизнеса к метрикам

Бизнес говорит: "Нам нужно лучше рекомендовать товары". ML Engineer слышит: "Задача ранжирования. Метрика — NDCG@10. Нужно учитывать cold start для новых пользователей".

Примеры перевода:

Бизнес-задача ML-задача Метрика

Уменьшить отток Бинарная классификация ROC-AUC, Recall

Рекомендовать товары Ранжирование NDCG, MAP

Предсказать спрос Регрессия MAE, RMSE

Найти мошенников Аномалия detection Precision@K

Сегментировать клиентов Кластеризация Silhouette Score

Ключевой принцип: метрика должна быть связана с бизнес-ценностью. Если вы оптимизируете Accuracy, а бизнес теряет деньги на False Negative — вы решаете не ту задачу.

Глава 2. Математический минимум для инженера

2.1. Линейная алгебра для тензорных вычислений

Нейросеть — это последовательность матричных умножений. Понимание линейной алгебры — не академическое упражнение, а необходимость.

Векторы и матрицы:

```
python
```

```
import numpy as np
```

```
# Вектор-столбец
```

```
v = np.array([[1], [2], [3]]) # shape (3, 1)
```

```
# Матрица
```

```
W = np.array([[1, 2], [3, 4], [5, 6]]) # shape (3, 2)
```

Умножение матрицы на вектор

result = W.T @ v # shape (2, 1)

Тензоры:

Тензор — это обобщение векторов и матриц на произвольное число измерений.

Скаляр: 0D тензор

Вектор: 1D тензор

Матрица: 2D тензор

Батч изображений: 4D тензор (batch, channels, height, width)

Операции, которые нужно знать:

Скалярное произведение: мера похожести векторов.

Матричное умножение: основа линейных слоев.

Транспонирование: изменение порядка осей.

Broadcasting: автоматическое расширение размерностей.

2.2. Теория вероятностей и статистика

Почему модели ошибаются?

Потому что данные — это выборка из распределения. Мы обучаем модель на одной выборке, а применяем на другой.

Ключевые концепции:

Распределение: функция, описывающая вероятность значений.

Математическое ожидание: среднее значение.

Дисперсия: мера разброса.

Ковариация: мера совместной изменчивости.

Байесовский подход:

$$P(\text{гипотеза} \mid \text{данные}) = P(\text{данные} \mid \text{гипотеза}) * P(\text{гипотеза}) / P(\text{данные})$$

Это основа для понимания того, как модели обновляют свои убеждения.

2.3. Градиентный спуск: от формулы до PyTorch

Математика:

Мы хотим минимизировать функцию потерь $L(\theta)$. Гради-

ентный спуск обновляет параметры:

$$\theta_{\{t+1\}} = \theta_t - \eta * \nabla L(\theta_t)$$

где η — learning rate.

В PyTorch:

```
python
import torch
import torch.nn as nn

model = nn.Linear(10, 1)
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
criterion = nn.MSELoss()

# Forward pass
y_pred = model(X)
loss = criterion(y_pred, y_true)

# Backward pass
optimizer.zero_grad()
loss.backward()
optimizer.step()
Виды градиентного спуска:
```

Batch GD: использует все данные. Медленно, но стабильно.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.