

Машинное обучение и нейросетевые модели

Лекция 2. Байесовские методы в машинном обучении

Лектор: Кравченя Павел Дмитриевич

Волгоград 2025

План лекции

1. Основные понятия классического машинного обучения.
 2. Классический и байесовский подходы к машинному обучению.
 3. Принципы максимальных правдоподобия и апостериорной вероятности.
 4. Частотный и байесовский подходы к описанию событий в ML.
 5. Основная проблема байесовского подхода.
 6. Оценка интегралов методами Монте-Карло.
 7. Понятия и виды семплирования случайных величин. Forward sampling.
 8. Фреймворк вероятностного программирования Pyro.
 9. Реализация вероятностной модели с помощью Pyro.
 10. Реализация сторонних эффектов в Pyro. Понятие «трасс».
-

- **Объектом** называется сущность, к которой применяется алгоритм ML.
- **Признаками** называются некоторые характеристики объекта, набор которых определяет объект в задачах машинного обучения.
- Набор объектов и соответствующих им признаков представляет собой **матрицу «объекты-признаки»**:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix} \in \mathbb{X}, \quad \mathbb{X} = \mathbb{R}^{n \times m}$$

где n – количество объектов, m – количество признаков каждого объекта.

- В задачах обучения «с учителем» определяется вектор **меток** для множества объектов:

$$\mathbf{t} = (t_1 \quad t_2 \quad \cdots \quad t_n)^T \in \mathbb{T}, \quad \mathbb{T} = \mathbb{R}^n \text{ или } \mathbb{T} = \{c_0, c_1, \dots, c_K\}^n$$

- Датасет, для всех объектов которого определены как признаки, так и метки, называется **обучающей выборкой**.
- Предсказываемый системой машинного обучения результат называется **целевой переменной**:

$$\mathbf{y} = (y_1 \ y_2 \ \dots \ y_n)^T \in \mathbb{Y}, \quad \mathbb{Y} = \mathbb{R}^n \text{ или } \mathbb{Y} = \{c_0, c_1, \dots, c_K\}^n$$

- **Алгоритмом**, или моделью ML, называется функция: $a: \mathbb{X} \rightarrow \mathbb{Y}$.
- Как правило, при решении задачи задают определенный класс функций модели с точностью до параметров $\mathbf{w} = (w_0, w_1, \dots, w_m)$: $\mathbf{y} = a(\mathbf{X}, \mathbf{w})$.
- Качество работы модели на одном объекте из обучающей выборки оценивают с помощью **функции ошибки**: $L(y, t) \in \mathbb{R}_+$.
- Функция ошибки *равна нулю*, если предсказанное значение *равно* метке. Чем *сильнее* отличаются y и t , тем *больше* значение функции ошибки.

- Качество работы модели на всей обучающей выборке размера n определяют с использованием **эмпирического риска**:

$$Q(\mathbf{X}, \mathbf{t}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n L[a(\mathbf{x}_i, \mathbf{w}), t_i].$$

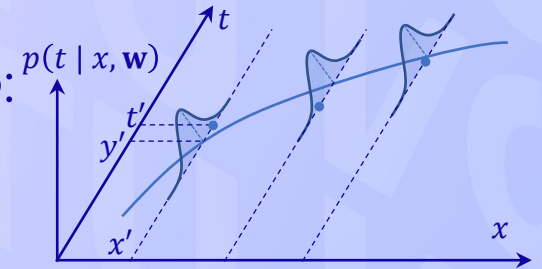
- Принцип **минимума эмпирического риска** выражает факт того, что лучшим образом обученная модель должна соответствовать минимально возможному значению эмпирического риска.
- **Обучить модель** с точки зрения классического машинного обучения означает получить значения весов модели, при которых она достигает минимального значения эмпирического риска:

$$\mathbf{w}_{opt} = \arg \min_{\mathbf{w} \in W} [Q(\mathbf{X}, \mathbf{t}, \mathbf{w})] = \arg \min_{\mathbf{w} \in W} \left[\sum_{i=1}^n L[a(\mathbf{x}_i, \mathbf{w}), t_i] \right].$$

Классический подход к ML	Байесовский подход к ML
<p>1. <u>Модель</u>:</p> $\mathbf{y} = \mathbf{X}\mathbf{w} \Leftrightarrow y_i = w_n x_{i,n} + w_{n-1} x_{i,n-1} + \dots + w_1 x_{i,1} + w_0 \quad \forall i \in [1..n]$ <p>2. <u>Датасеты</u>: <i>train</i>: $\mathcal{D} = (\mathbf{X}, \mathbf{t})$, <i>test</i>: $\mathcal{D}^{pr} = (\mathbf{X}^{pr})$</p> <p>3. <u>Обучение</u>: минимизация эмпирического риска:</p> $\mathbf{w}_{opt} = \arg \min_{\mathbf{w} \in \mathbb{W}} \left[\sum_{i=1}^n L[a(x_i, \mathbf{w}), t_i] \right]$	

- Пусть модель имеет ошибку, распределенную *нормально*:

$$\mathbf{t} = \mathcal{N}(\mathbf{y}, \sigma \mathbb{I}) \Rightarrow p(\mathbf{t} | \mathbf{X}, \mathbf{t}, \mathbf{w}) = \frac{1}{2\pi\sqrt{\sigma}} \cdot e^{-\frac{[a(\mathbf{X}, \mathbf{w}) - \mathbf{t}]^2}{2\sigma^2}}$$



- Запишем правдоподобие модели для датасета $\mathcal{D} = (\mathbf{X}, \mathbf{t}) = \{\mathbf{x}_i, t_i\}_{i=1}^m$:

$$p(\mathcal{D} | \mathbf{w}) = p(\mathbf{t} | \mathbf{X}, \mathbf{w}) = \prod_{i=1}^n p(t_i | \mathbf{x}_i, \mathbf{w}) = \left(\frac{1}{2\pi\sqrt{\sigma}} \right)^n \cdot \prod_{i=1}^n e^{-\frac{[a(x_i, \mathbf{w}) - t_i]^2}{2\sigma^2}}$$

- Запишем **логарифм правдоподобия** с целью избавиться от произведений:

$$\log[p(\mathcal{D} | \mathbf{w})] = n \cdot \log\left(\frac{1}{2\pi\sqrt{\sigma}}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^n [a(x_i, \mathbf{w}) - t_i]^2$$

$$\log[p(\mathcal{D} | \mathbf{w})] \rightarrow \max_{\mathbf{w} \in \mathbf{W}} \Leftrightarrow \frac{1}{n} \cdot \sum_{i=1}^n [a(x_i, \mathbf{w}) - t_i]^2 \rightarrow \min_{\mathbf{w} \in \mathbf{W}}$$

Классический подход к ML	Байесовский подход к ML
<p>1. <u>Модель</u>:</p> $\mathbf{y} = \mathbf{X}\mathbf{w} \Leftrightarrow y_i = w_n x_{i,n} + w_{n-1} x_{i,n-1} + \dots + w_1 x_{i,1} + w_0$ $\forall i \in [1..n]$ <p>2. <u>Датасеты</u>: <i>train</i>: $\mathcal{D} = (\mathbf{X}, \mathbf{t})$, <i>test</i>: $\mathcal{D}^{pr} = (\mathbf{X}^{pr})$</p> <p>3. <u>Обучение</u>: минимизация эмпирического риска:</p> $\mathbf{w}_{opt} = \arg \min_{\mathbf{w} \in \mathbb{W}} \left[\sum_{i=1}^n L[a(x_i, \mathbf{w}), t_i] \right]$ <p>эквивалентна <u>максимизации правдоподобия</u>:</p> $\mathbf{w}_{opt} = \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\prod_{i=1}^n p(t_i x_i, \mathbf{w}) \right] = \arg \max_{\mathbf{w} \in \mathbb{W}} (P(\mathbf{t} \mathbf{X}, \mathbf{w}))$ <p>Решается аналитически / градиентными методами.</p> <p>4. <u>Предсказание</u> по обученной модели:</p> $\mathbf{y}^{pr} = a(\mathbf{X}^{pr}, \mathbf{w}_{opt}) = \mathbf{X}^{pr} \mathbf{w}_{opt}$	

1. Определимся, какие величины, входящие в модель, будут **случайными**, а какие – **детерминированными**.

- ✓ Случайные величины имеют **распределение**, в то время как детерминированные представлены **конкретным значением**.
- ✓ Нужно определить, какие величины в модели мы будем моделировать. Далее, нужно понять, для каких величин в модели важно и нужно распределение, и мы готовы его определять, а для каких – достаточно численного значения.
- ✓ Случайные величины, конкретную реализацию которых можно получить из данных (датасета), будут **наблюдаемыми**, остальные – **латентными**.
- ✓ В модели должны присутствовать как наблюдаемые, так и латентные величины. Чаще всего, латентными величинами являются определяемые параметры модели.

Пример: линейная регрессия.

- ✓ **Признаки X** считаются известными детерминированными значениями, которые служат входными данными для модели. Их обычно не моделируют, поскольку они заданы в датасете.
 - Но если признак пропущен в модели, и сама модель используется для его оценки, то его потребуется промоделировать. Тогда, признак становится случайной величиной.

- ✓ **Целевая переменная y** определена только для обучающей выборки, но на этапе предсказаний её нужно получить из модели. Моделируем её случайной величиной, которая на этапе обучения будет наблюдаемой, а на этапе предсказаний – латентной.
 - ✓ **Параметры регрессии w** связывают целевую переменную с признаками и содержат некоторую степень неопределённости. Параметры регрессии требуется моделировать, рассматриваем их как латентные случайные величины.
2. Записываем **совместное распределение** случайных величин и выражаем его в общем виде через правдоподобие и априорные вероятности.

$$p(\mathbf{w}, \mathcal{D}) = p(\mathcal{D} | \mathbf{w}) \cdot p(\mathbf{w}).$$

3. **Конкретизируем** записанные правдоподобие и априорные вероятности для конкретной, рассматриваемой модели, учитываем её особенности.

Пример: линейная регрессия.

- ✓ Совместное распределение целевой переменной и весов запишется в виде:

$$p(\mathbf{w}, y, \mathbf{X}) = p(y | \mathbf{w}, \mathbf{X}) \cdot p(\mathbf{w}).$$

- ✓ Будем считать, что в серии экспериментов целевая переменная для каждого объекта **не зависит** от других объектов, т.е. условная плотность распределения целевой переменной факторизуется:

$$p(\mathbf{y} | \mathbf{w}, \mathbf{X}) = \prod_{i=1}^n p(y_i | \mathbf{w}, \mathbf{x}_i).$$

- ✓ Здесь случайные величины y_i являются **локальными**, поскольку относятся к конкретным наблюдениям, а величина \mathbf{w} – **глобальная**, поскольку описывает характеристики всего датасета.
- ✓ Положим, что каждая целевая переменная имеет нормальное распределение со средним, которое определяется как линейная комбинация весов регрессии и признаков, и параметром – среднеквадратическим отклонением σ :

$$p(y_i | \mathbf{w}, \mathbf{x}_i) = \mathcal{N}(\mu_i, \sigma^2); \quad \mu_i = (\mathbf{X}\mathbf{w})_i = \sum_{j=1}^m w_j x_{ij} + w_0.$$

- ✓ Каждый из признаков независимо от других входит в модель; логично положить, что все веса регрессии для всех m признаков также **независимы** друг от друга:

$$p(\mathbf{w}) = \prod_{j=1}^m p(w_j) \cdot p(w_0).$$

- ✓ Теперь нужно задать априорные распределения весов модели $p(w_0), p(w_1), \dots, p(w_m)$. Априорные распределения задают наши изначальные представления о распределении весов.
- ✓ Априорные распределения могут быть информативными и неинформативными.
- ✓ **Информативное** распределение отражает сильные предварительные убеждения о значении параметра. Например, мы из личного опыта знаем, какие примерно значения может принимать вес линейной регрессии в конкретной задаче.
- ✓ **Неинформативное** распределение отражает минимальные предварительные убеждения о значении параметра. Например, полагает одинаковую вероятность весов в некоторой области.
- ✓ Пусть, наши предварительные знания позволяют нам использовать только неинформативное распределение, например:

$$p(w_j) = \mathcal{N}(0, \sigma_0^2) \quad \forall j \in [0, m], \quad \sigma_0 \in \mathbb{R}_+.$$

- ✓ Таким образом, мы задали **байесовскую модель линейной регрессии**:

$$p(\mathbf{w}, \mathbf{y}, \mathbf{X}) = \prod_{i=1}^n p(y_i | \mathbf{w}, \mathbf{x}_i) \cdot \prod_{j=1}^m p(w_j) \cdot p(w_0).$$

$$p(y_i | \mathbf{w}, \mathbf{x}_i) = \mathcal{N}(\mu_i, \sigma^2);$$

$$\mu_i = \sum_{j=1}^m w_j x_{ij} + w_0;$$

$$p(w_j) = \mathcal{N}(0, \sigma_0^2) \quad \forall j \in [0, m], \quad \sigma_0 \in \mathbb{R}_+.$$

- Определить байесовскую модель – это означает определить совместное распределение случайных величин, входящих в неё: $p(\mathbf{w}, \mathcal{D})$.
- В модели задаются априорные распределения латентных случайных величин, которые, после того, как мы пронаблюдали данные, изменятся в соответствии с **теоремой Байеса**:

$$p(\mathbf{w} | \mathcal{D}) = \frac{p(\mathcal{D} | \mathbf{w}) \cdot p(\mathbf{w})}{\int_{\mathcal{S}} p(\mathcal{D} | \mathbf{w}) \cdot p(\mathbf{w}) d\mathbf{w}}.$$

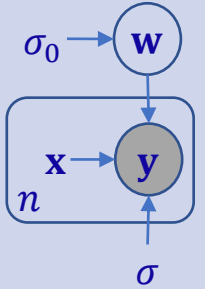
- Процесс вычисления апостериорных распределений латентных величин называется **байесовским инференсом**.
- Инференс относится к этапу обучения модели. Он предполагает получение распределений весов, с которыми модель с заданными правдоподобием и априорными распределениями весов, описывающих наши изначальные знания, эффективно описывает наблюдаемые данные.

- Как правило, задача **предсказания по байесовской модели** заключается в определении распределения случайных переменных, которые во время инференса были наблюдаемыми, а теперь стали латентными. Например, целевая переменная на тестовом множестве.
- Эта задача сводится к определению предиктивного распределения.
- **Предиктивным** называют распределение ненаблюдаемой переменной y^{pr} при условии наблюдаемых данных \mathcal{D}^{pr} и обучающих данных \mathcal{D} :

$$p(y^{pr} | \mathcal{D}^{pr}, \mathcal{D}) = \int_{\mathbf{w}} p(\mathcal{D}^{pr} | \mathbf{w}) \cdot p(\mathbf{w} | \mathcal{D}) d\mathbf{w} = \mathbb{E}_{\mathbf{w} \sim p(\mathbf{w} | \mathcal{D})} [p(\mathcal{D}^{pr} | \mathbf{w})].$$

Пример: линейная регрессия с правдоподобием $p(\mathbf{y} | \mathbf{w}, \mathbf{X})$ и апостериорным распределением весов $p(\mathbf{w} | \mathbf{y}, \mathbf{X})$:

$$p(y^{pr} | \mathbf{X}^{pr}, \mathbf{y}, \mathbf{X}) = \int_{\mathbf{w}} p(y^{pr} | \mathbf{X}^{pr}, \mathbf{w}) \cdot p(\mathbf{w} | \mathbf{y}, \mathbf{X}) d\mathbf{w} = \mathbb{E}_{\mathbf{w} \sim p(\mathbf{w} | \mathbf{y}, \mathbf{X})} [p(y^{pr} | \mathbf{X}^{pr}, \mathbf{w})].$$

Классический подход к ML	Байесовский подход к ML
<p>1. <u>Модель</u>:</p> $\mathbf{y} = \mathbf{X}\mathbf{w} \Leftrightarrow y_i = w_n x_{i,n} + w_{n-1} x_{i,n-1} + \dots + w_1 x_{i,1} + w_0 \quad \forall i \in [1..n]$ <p>2. <u>Датасеты</u>: <i>train</i>: $\mathcal{D} = (\mathbf{X}, \mathbf{t})$, <i>test</i>: $\mathcal{D}^{pr} = (\mathbf{X}^{pr})$</p> <p>3. <u>Обучение</u>: минимизация эмпирического риска:</p> $\mathbf{w}_{opt} = \arg \min_{\mathbf{w} \in \mathbb{W}} \left[\sum_{i=1}^n L[a(x_i, \mathbf{w}), t_i] \right]$ <p>эквивалентна <u>максимизации правдоподобия</u>:</p> $\mathbf{w}_{opt} = \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\prod_{i=1}^n p(t_i x_i, \mathbf{w}) \right] = \arg \max_{\mathbf{w} \in \mathbb{W}} (P(\mathbf{t} \mathbf{X}, \mathbf{w}))$ <p>Решается аналитически / градиентными методами.</p> <p>4. <u>Предсказание</u> по обученной модели:</p> $\mathbf{y}^{pr} = a(\mathbf{X}^{pr}, \mathbf{w}_{opt}) = \mathbf{X}^{pr} \mathbf{w}_{opt}$	<p>1. <u>Модель</u>:</p> $p(\mathbf{y}, \mathbf{X}, \mathbf{w}) = p(\mathbf{y} \mathbf{X}, \mathbf{w}) p(\mathbf{w}) = p(\mathbf{w}) \prod_{i=1}^n p(y_i \mathbf{x}_i, \mathbf{w})$ $p(w_j) = \mathcal{N}(0, \sigma_0^2),$ $p(y_i \mathbf{x}_i, \mathbf{w}) = \mathcal{N}(\mathbf{x}_i \cdot \mathbf{w}, \sigma^2) \quad \forall j \in [0..m]$  <p>2. <u>Датасеты</u>: <i>train</i>: $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, <i>test</i>: $\mathcal{D}^{pr} = (\mathbf{X}^{pr})$</p> <p>3. <u>Обучение</u>: <u>определение апостериорных вероятностей весов модели по теореме Байеса</u>:</p> $p(\mathbf{w} \mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y} \mathbf{X}, \mathbf{w}) p(\mathbf{w})}{\int p(\mathbf{y} \mathbf{X}, \mathbf{w}) p(\mathbf{w}) d\mathbf{w}}$ <p>Решается процедурой байесовского инференса.</p> <p>4. <u>Предсказание</u> по обученной модели:</p> $p(\mathbf{y}^{pr} \mathbf{X}^{pr}, \mathbf{y}, \mathbf{X}) = \int p(\mathbf{y}^{pr} \mathbf{X}^{pr}, \mathbf{w}) p(\mathbf{w} \mathbf{y}, \mathbf{X}) d\mathbf{w}$

- Вычислим вместо апостериорного распределения весов их значения, при которых достигается максимум распределения:

$$\mathbf{w}_{opt}^{MAP} = \arg \max_{\mathbf{w} \in \mathcal{W}} [p(\mathbf{w} | \mathbf{y}, \mathbf{X})] = \arg \max_{\mathbf{w} \in \mathcal{W}} [p(\mathbf{y} | \mathbf{X}, \mathbf{w})p(\mathbf{w})]$$

- Получили точечную оценку из принципа **максимального апостериорного распределения**. В ряде случаев, оценка может быть *нерепрезентативной*.
- Пусть мы не знаем априорное распределение весов. Предположим, что веса распределены равномерно, т.е. $w_i \sim \mathcal{U}(a, b)$. Тогда принцип MAP примет вид:

$$\mathbf{w}_{opt}^{MLE} = \arg \max_{\mathbf{w} \in \mathcal{W}} \left[\frac{1}{b-a} \cdot p(\mathbf{y} | \mathbf{X}, \mathbf{w}) \right] = \arg \max_{\mathbf{w} \in \mathcal{W}} [p(\mathbf{y} | \mathbf{X}, \mathbf{w})]$$

- Получили принцип **максимального правдоподобия**, применяющийся при точечной оценке весов в *классическом машинном обучении*.

- Рассмотрим записанную ранее модель линейной регрессии с априорными распределениями весов и правдоподобием в виде:

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{i=1}^n \mathcal{N}(y_i; \mathbf{x}_i \cdot \mathbf{w}, \sigma^2), \quad p(\mathbf{w}) = \prod_{j=1}^m \mathcal{N}(w_j; 0, \sigma_0).$$

- Тогда оценка максимального правдоподобия выразится таким образом:

$$\begin{aligned} \mathbf{w}_{opt}^{MLE} &= \arg \max_{\mathbf{w} \in \mathbb{W}} [p(\mathbf{w}) \cdot p(\mathbf{y} \mid \mathbf{X}, \mathbf{w})] = \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\prod_{j=1}^m \mathcal{N}(w_j; 0, \sigma_0) \cdot \prod_{i=1}^n \mathcal{N}(y_i; \mathbf{x}_i \cdot \mathbf{w}, \sigma^2) \right] = \\ &= \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\log \left(\prod_{j=1}^m \mathcal{N}(w_j; 0, \sigma_0) \cdot \prod_{i=1}^n \mathcal{N}(y_i; \mathbf{x}_i \cdot \mathbf{w}, \sigma^2) \right) \right] = \end{aligned}$$

$$\begin{aligned} &= \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\log \left(\prod_{j=1}^m \mathcal{N}(w_j; 0, \sigma_0) \right) + \log \left(\prod_{i=1}^n \mathcal{N}(y_i; \mathbf{x}_i \cdot \mathbf{w}, \sigma^2) \right) \right] = \\ &= \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\sum_{j=1}^m \log \mathcal{N}(w_j; 0, \sigma_0) + \sum_{i=1}^n \log \mathcal{N}(y_i; \mathbf{x}_i \cdot \mathbf{w}, \sigma^2) \right] = \\ &= \arg \max_{\mathbf{w} \in \mathbb{W}} \left[\sum_{j=1}^m \log \left(\frac{1}{\sqrt{2\pi}\sigma_0} \cdot e^{-\frac{w_j^2}{2\sigma_0}} \right) + \sum_{i=1}^n \log \left(\frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma}} \right) \right] = \\ &= \arg \max_{\mathbf{w} \in \mathbb{W}} \left[m \cdot \log \left(\frac{1}{\sqrt{2\pi}\sigma_0} \right) - \frac{1}{2\sigma_0} \sum_{j=1}^m w_j^2 + n \cdot \log \left(\frac{1}{\sqrt{2\pi}\sigma} \right) - \frac{1}{2\sigma} \sum_{i=1}^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2 \right] = \end{aligned}$$

$$\begin{aligned} &= \arg \min_{\mathbf{w} \in \mathbb{W}} \left[\frac{1}{2\sigma_0} \sum_{j=1}^m w_j^2 + \frac{1}{2\sigma} \sum_{i=1}^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2 \right] = \arg \min_{\mathbf{w} \in \mathbb{W}} \left[\frac{1}{2\sigma} \cdot \left(\frac{\sigma}{\sigma_0} \sum_{j=1}^m w_j^2 + \sum_{i=1}^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2 \right) \right] = \\ &= \arg \min_{\mathbf{w} \in \mathbb{W}} \left[\frac{\sigma}{\sigma_0} \sum_{j=1}^m w_j^2 + \sum_{i=1}^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2 \right] = \arg \min_{\mathbf{w} \in \mathbb{W}} \left[L_{\text{reg}}(\mathbf{w}) + \sum_{i=1}^n L[a(\mathbf{x}_i, \mathbf{w}), y_i] \right]. \end{aligned}$$

- Получили L_2 -регуляризацию: $L_{\text{reg}}(\mathbf{w}) = \alpha \cdot \sum_{j=1}^m w_j^2$, $\alpha = \frac{\sigma}{\sigma_0}$.
- Таким образом, заданное априорное распределение в байесовской модели имеет смысл регуляризации в классической модели, причём семейство распределения задаёт вид регуляризационного слагаемого.
- Можно показать, что априорное распределение весов по закону Лапласа эквивалентно L_1 -регуляризации.

	Частотный подход	Байесовский подход
Интерпретация случайности в модели	Объективная неопределенность	Субъективное незнание
Величины в модели	Как случайные, так и детерминированные	Все случайные
Метод вывода неизвестных величин	Максимальное правдоподобие: $\mathbf{w}_{opt} = \operatorname{argmax}_{\mathbf{w} \in \mathcal{W}} [p(\mathcal{D} \mathbf{w})]$	Теорема Байеса: $p(\mathbf{w} \mathcal{D}) = \frac{p(\mathcal{D} \mathbf{w})p(\mathcal{D})}{\int p(\mathcal{D} \mathbf{w})p(\mathbf{x})}$
Оценки величин	Точечные: \mathbf{w}_{opt}	Апостериорное распределение: $p(\mathbf{w} \mathcal{D})$
Применимость моделей	При больших объёмах выборки: $n \gg d$ n – количество объектов выборки, d – количество весов модели.	При любых объёмах выборки: $\forall n$

- Байесовский подход к ML предполагает расчет следующих выражений:

$$p(\mathbf{w} | \mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y} | \mathbf{X}, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y} | \mathbf{X}, \mathbf{w})p(\mathbf{w})d\mathbf{w}}$$

$$p(\mathbf{y}^{\text{pr}} | \mathbf{X}^{\text{pr}}, \mathbf{y}, \mathbf{X}) = \int p(\mathbf{y}^{\text{pr}} | \mathbf{X}^{\text{pr}}, \mathbf{w})p(\mathbf{w} | \mathbf{y}, \mathbf{X})d\mathbf{w}$$

- Основную проблему при выполнении инференса представляет собой вычисление значения *интеграла*.
- Данный интеграл считается «*intractable*»:
 - ✓ **Аналитически** данный интеграл можно рассчитать только в некоторых частных случаях.
 - ✓ **Численно** данный интеграл нельзя вычислить с разумными вычислительными затратами для большинства реальных моделей.
- Для вычисления или оценки значения данного интеграла приходится использовать специальные методы.

- Аналитическое вычисление апостериорного распределения в теореме Байеса возможно только в очень ограниченном числе случаев, когда правдоподобие и априорное распределение имеют специальные формы: так называемых сопряженных распределений.
- Если априорное распределение $p(\mathbf{w})$ является **сопряженным** к функции правдоподобия $p(\mathcal{D} | \mathbf{w})$, то апостериорное распределение $p(\mathbf{w} | \mathcal{D})$ принадлежит к тому же семейству распределений, что и априорное распределение.
- Это означает, что апостериорное распределение может быть получено путем **обновления параметров** априорного распределения на основе наблюдаемых данных. Расчет сложного интеграла при этом не требуется.
- Однако, сопряженные априорные распределения доступны только для ограниченного числа моделей, что ограничивает их применение.

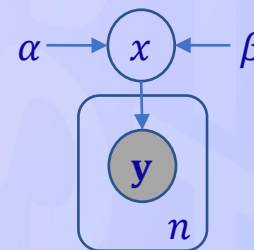
Априорное распределение	Функция правдоподобия	Апостериорное распределение
Бета: $Beta(x; \alpha, \beta)$	Бернулли: $\mathcal{B}(\mathbf{x}; p) = \prod_{i=1}^n p^{x_i} \cdot (1 - p)^{1-x_i}$	$\alpha' = \alpha + \sum_{i=1}^n x_i, \quad \beta' = \beta + n - \sum_{i=1}^n x_i.$ <p>α увеличивается на число успехов, β увеличивается на число неудач.</p>
Дирихле: $Dir(\mathbf{x}; \boldsymbol{\alpha})$	Категориальное: $Cat(x; k, \mathbf{p}) = \prod_{i=1}^n \prod_{j=1}^k p^{x_{ik}}$	$\alpha_j' = \alpha_j + n_k$ <p>Параметры обновляются добавлением числа наблюдений для каждого j.</p>
Гамма: $\mathcal{G}(x; k, \theta)$	Пуассона: $\mathcal{Pois}(x; \lambda) = \prod_{i=1}^n \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}$	$k' = k + \sum_{i=1}^n x_i, \quad \theta' = \frac{\theta}{n\theta + 1}.$
Нормальное: $\mathcal{N}(x; \mu_0, \sigma_0^2)$	Нормальное с известной σ^2 : $\mathcal{N}(x; \mu, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x_i - \mu)^2}{2\sigma^2}}$	$\mu_0' = \frac{\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^n x_i}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}, \quad \sigma_0'^2 = \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}.$

- Пусть нам дана некоторая монета, и нам требуется определить, является ли она симметричной.
- Для этого её подбросили 10 раз и зафиксировали результаты. Они приведены в таблице.

№ броска	Результат
1	Орёл
2	Решка
3	Орёл
4	Решка
5	Решка
6	Решка
7	Решка
8	Орёл
9	Решка
10	Решка

- Построим байесовскую модель эксперимента:

- ✓ x – вероятность выпадения «орла» в одном эксперименте;
- ✓ y – вектор результатов всех экспериментов (из таблицы);
- ✓ n – количество экспериментов;
- ✓ $\alpha = 2, \beta = 2$ – гиперпараметры.



$$p(x, y) = p(y | x) \cdot p(x); \quad p(y | x) = \prod_{i=1}^n p(y_i | x); \quad p(x) = \text{Beta}(x; \alpha = 2, \beta = 2).$$

- И байесовский вывод: $p(x | y) \propto \text{B}(y_i; x) \cdot \text{Beta}(x; \alpha, \beta) \Rightarrow p(x | y) = \text{Beta}(x; \alpha', \beta')$.

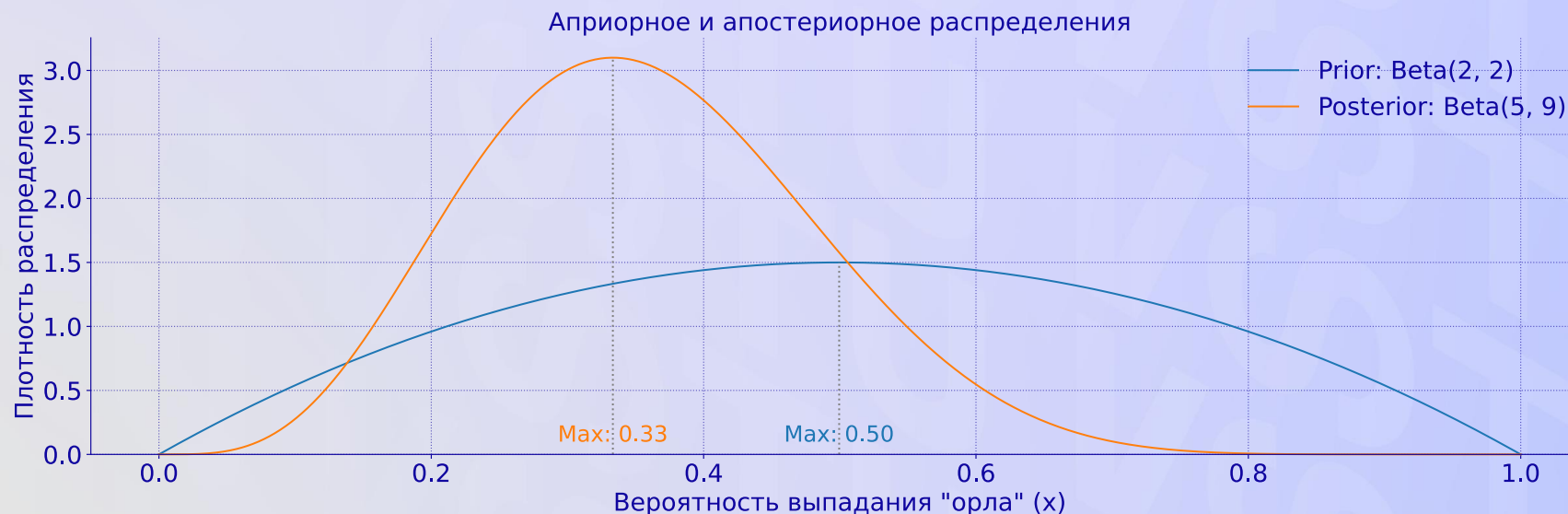
$$\alpha' = \alpha + \sum_{i=1}^n [y_i = \text{"Орёл"}] = 2 + 3 = 5; \quad \beta' = \beta + \sum_{i=1}^n [y_i = \text{"Решка"}] = 2 + 7 = 9;$$

$$p(x | y) = \text{Beta}(x; 5, 9).$$

- Итак, функция правдоподобия, априорное и апостериорное распределения приняли вид:

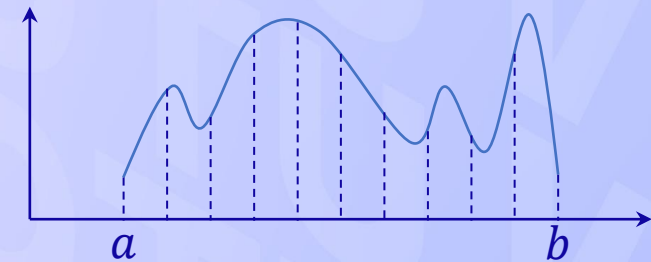
$$p(x) = \text{Beta}(x; 1, 1); \quad p(x | y) = \text{Beta}(x; 4, 8). \quad p(y | x) = \prod_{i=1}^n B(y_i; x).$$

- Видно, что априорное распределение сместилось влево.



№ броска	Результат
1	Орёл
2	Решка
3	Орёл
4	Решка
5	Решка
6	Решка
7	Решка
8	Орёл
9	Решка
10	Решка

- **Численное интегрирование** (например, метод трапеций, метод Симпсона) может быть использовано для аппроксимации интеграла.
- Если в одномерном случае область интегрирования разделяется на 10 областей, то в двумерном – та же *точность дискретизации* достигается при 100 областях. Если размерность равна D , то потребуется 10^D областей.
- Сложность численного интегрирования экспоненциально возрастает с увеличением размерности пространства параметров (**проклятие размерности**). Даже для умеренных размерностей, требуемая вычислительная мощность может быть огромной.

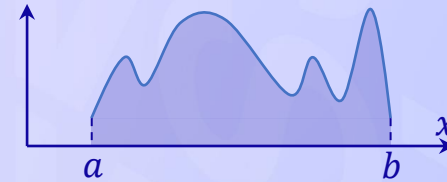


- Современный подход к оценке рассмотренных интегралов предполагает использование **методов Монте-Карло**.
- Методами Монте-Карло считается группа численных методов, использующих случайные величины для оценки характеристик модели.
- Методы получили своё *название* от административной территории княжества Монако, которая славится своими *казино*.
- Можно показать, что ошибка метода Монте-Карло при интегрировании равна $O\left(\frac{1}{\sqrt{N}}\right)$ при любой размерности величин, в то время как ошибка численных методов равна $O\left(\frac{1}{D\sqrt{N}}\right)$ при размерности D .
- Это делает методы Монте-Карло привлекательными для решения задач высокой размерности.

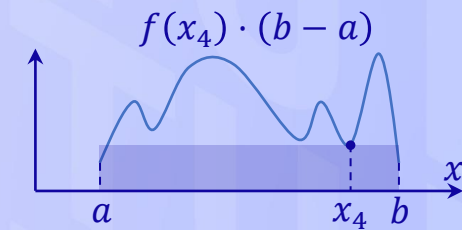
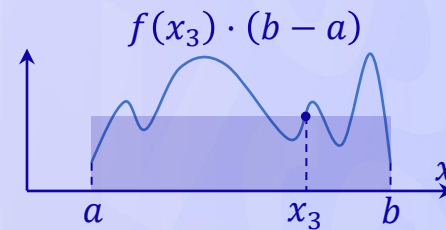
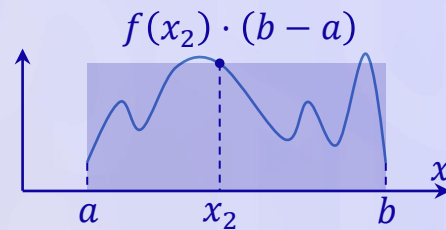
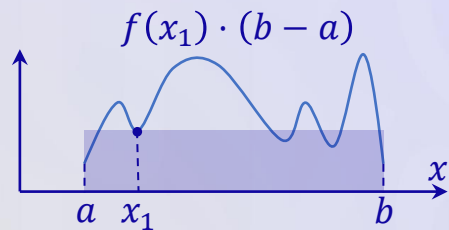


- Пусть требуется рассчитать интеграл вида (площадь под кривой $f(x)$):

$$F = \int_a^b f(x) dx$$



- Выберем равномерно случайно некоторые $x_i \in [a, b]$, рассчитаем значения функции f в них и усредним полученные значения:



$$\frac{1}{4} \cdot \left(\text{rectangle}_1 + \text{rectangle}_2 + \text{rectangle}_3 + \text{rectangle}_4 \right) \approx \text{area under } f(x)$$

- Если непосредственное вычисление некоторой величины затруднено или невозможно, для её оценки можно воспользоваться эстиматором.
- **Эстиматором** называется функция от выборки, которая используется для оценки неизвестной величины, связанной с генеральной совокупностью.
- Если эстиматор использует случайные числа, он называется эстиматором Монте-Карло.
- Если математическое ожидание эстиматора равно истинному значению оцениваемой величины, эстиматор называется **несмещённым**, иначе – смещённым. Смещение показывает систематическую ошибку эстиматора.
- **Мерой точности** эстиматора является его дисперсия. Чем *меньше* дисперсия, тем *точнее* (в смысле *разброса*) эстиматор.

- Пусть требуется рассчитать интеграл вида:

$$F = \int_a^b f(x)dx.$$

- Будем рассматривать величину X как случайную: $X \sim p(x)$
- Тогда **несмещенная** оценка значения интеграла методом Монте-Карло:

$$\hat{F}_N = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}.$$

$$\mathbb{E}[\hat{F}_N] = \mathbb{E}\left[\frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}\right] = \frac{1}{N} \sum_{i=1}^N \mathbb{E}\left[\frac{f(x_i)}{p(x_i)}\right] = \frac{1}{N} \sum_{i=1}^N \int_a^b \frac{f(x)}{p(x)} p(x) dx = \frac{1}{N} \sum_{i=1}^N \int_a^b f(x) dx = \int_a^b f(x) dx = F.$$

- Чаще всего в данном курсе потребуется вычислять интегралы вида:

$$\mathbb{E}_{p(x)}[f(x)] = \int_a^b f(x)p(x)dx.$$

- В таком случае, эстиматор Монте-Карло \hat{E}_N для подобного интеграла:

$$\mathbb{E}_{p(x)}[f(x)] \approx \hat{E}_N = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)p(x_i)}{p(x_i)} = \frac{1}{N} \sum_{i=1}^N f(x_i), \quad X \sim p(x), \quad x_i \in [a, b].$$

- Данный способ оценки матожидания по области Ω хорошо обобщается на **многомерный случай**:

$$\hat{E}_N = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i), \quad \mathbf{X} \sim p(\mathbf{x}), \quad \mathbf{x}_i \in \Omega.$$

- Для вычисления данного эстиматора нужно уметь семплировать из $p(\mathbf{x})$.

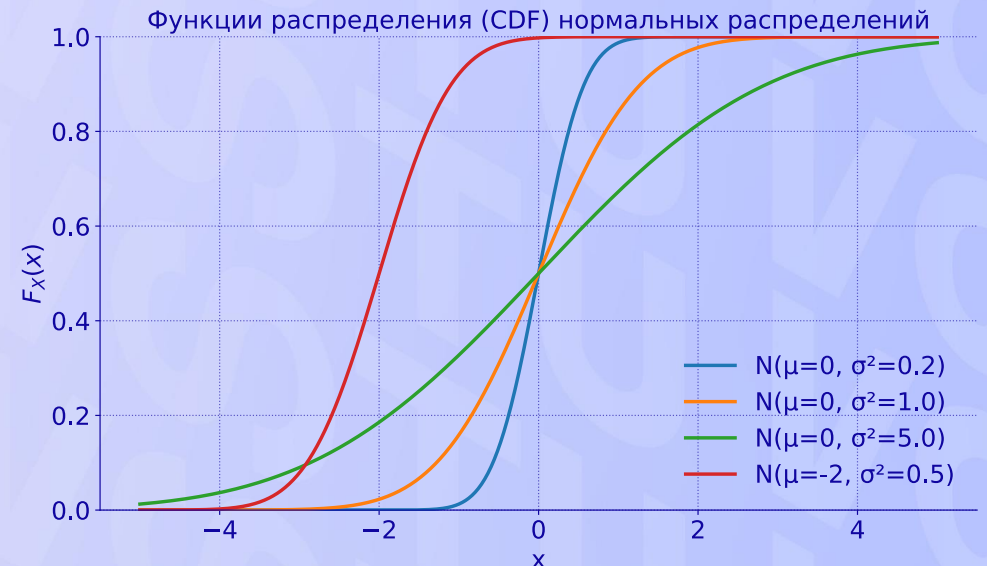
- Под **семплированием** понимают процесс генерации независимых и одинаково распределенных (i.i.d.) случайных величин из заданного вероятностного распределения. Эти сгенерированные значения образуют случайную выборку: $X \sim p(x)$.
- Обычно семплирование из *равномерного распределения* получают с использованием генератора псевдослучайных чисел (например, *Mersenne twister*), реализованного в виде вычисления рекуррентного соотношения: $x_{n+1} = g(x_n)$, $x_0 = \text{const}$ (*seed*), чтобы $\{x_1, x_2, \dots, x_n\}$ было выборкой из $\mathcal{U}(0, 1)$ с достаточной точностью.
- Семплирование из других распределений часто реализуется на основе стандартизированного равномерного распределения $\mathcal{U}(0, 1)$.

- Функция распределения $F_X(x)$ (Cumulative Distribution Function (CDF)) – функция, характеризующая распределение случайной величины X , равная вероятности того, что случайная величина примет значение $< x$:

$$F_X(x) = P(X \leq x).$$

- Свойства функции распределения:

- ✓ $D[F_X(x)] \in (-\infty; +\infty)$;
- ✓ $E[F_X(x)] \in [0; 1]$;
- ✓ $\lim_{x \rightarrow \infty} F_X(x) = 1$;
- ✓ $\lim_{x \rightarrow -\infty} F_X(x) = 0$;
- ✓ $F_X(x)$ – неубывающая на $(-\infty; +\infty)$.



- Из курса *теории вероятностей* известно, что:

$$F_X(x) = \int_{-\infty}^x p(x)dx.$$

- Для равномерной случайной величины:

$$p(x) = \begin{cases} 0, & \text{if } x > b; \\ C, & \text{if } a < x \leq b; \\ 0, & \text{if } x \leq a. \end{cases}$$

- Согласно условию нормировки: $\int p(x) = 1 \Rightarrow C = \frac{1}{b-a}$.
- Тогда функция равномерного распределения:

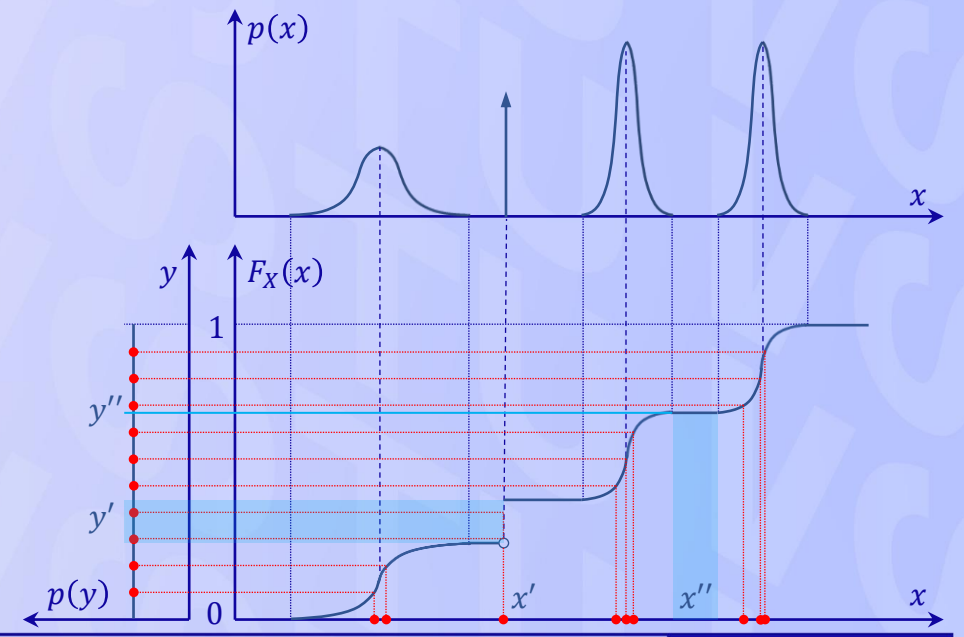
$$F_X(x) = \frac{1}{b-a} \int_{-\infty}^x p(x)dx = \frac{x-a}{b-a}, \quad \boxed{F_{X \sim U(0,1)}(x) = x} \text{ при } 0 \leq x \leq 1.$$

Теорема об обратной функции распределения

- Пусть $y = F_X(x)$, где $F_X(x)$ – функция распределения некоторой случайной величины X , и $F_X^{-1}(y)$ – обратная функция распределения:

$$F_X^{-1}(y) = \inf \{x \in \mathbb{R} : F_X(x) \geq y\}, \quad (0 \leq y \leq 1).$$

- Теорема.** Если случайная величина $Y \sim \mathcal{U}(0, 1)$, то случайная величина $X = F_X^{-1}(Y)$, имеет функцию распределения F_X , т.е., $P(X \leq x) = F_X(x) \quad \forall x \in \mathbb{R}$.
- Условие $Y \sim \mathcal{U}(0, 1)$ и преобразование $X = F_X^{-1}(Y)$ – это метод генерации X с заданным распределением F_X .



- **Доказательство.** Пусть $Y \sim \mathcal{U}(0, 1)$. Рассмотрим функцию распределения случайной величины $X = F^{-1}(Y)$:

$$F_X(x) = P(X \leq x) = P(F_X^{-1}(Y) \leq x).$$

- Покажем, что:

$$\boxed{F_X^{-1}(Y) \leq x} \Leftrightarrow \boxed{Y \leq F_X(x)}.$$

1. Пусть $F^{-1}(Y) \leq x$. По определению обратной функции распределения:

$$F_X^{-1}(Y) = \inf \{ \xi \in \mathbb{R} : F_X(\xi) \geq Y \} \Rightarrow x \geq \inf \{ \xi \in \mathbb{R} : F_X(\xi) \geq Y \}.$$

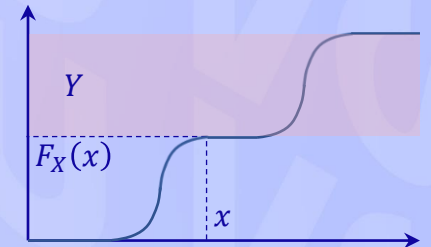
Так как $F_X(x)$ – неубывающая и непрерывная справа, то:

$$F_X(\xi_1) \geq F_X(\xi_2) \quad \forall \xi_1, \xi_2 : \xi_1 > \xi_2.$$

Предположим, что $Y > F_X(x)$. Тогда:

$$\{ \xi \in \mathbb{R} : F_X(\xi) \geq Y \} \subset \{ \xi \in \mathbb{R} : F_X(\xi) \geq F_X(x) \} \Rightarrow \inf \{ \xi \in \mathbb{R} : F_X(\xi) \geq Y \} > x \Rightarrow F_X^{-1}(Y) > x.$$

Получили противоречие. Значит, $F_X^{-1}(Y) \leq x \Rightarrow Y \leq F_X(x)$.



2. Пусть $Y \leq F_X(x)$. Значит, $F_X(x) \geq Y$, и в таком случае:

$$x \in \{\xi \in \mathbb{R} : F_X(\xi) \geq Y\}.$$

Тогда:

$$\inf \{\xi \in \mathbb{R} : F_X(\xi) \geq Y\} \leq x.$$

И в соответствии с определением обратной функции распределения:

$$F_X^{-1}(Y) = \inf \{\xi \in \mathbb{R} : F_X(\xi) \geq Y\} \Rightarrow x \geq F_X^{-1}(Y).$$

Получили, что $Y \leq F_X(x) \Rightarrow F_X^{-1}(Y) \leq x$.

- Таким образом:
$$\begin{cases} F_X^{-1}(Y) \leq x \Rightarrow Y \leq F_X(x) \\ Y \leq F_X(x) \Rightarrow F_X^{-1}(Y) \leq x \end{cases} \Leftrightarrow (F_X^{-1}(Y) \leq x \Leftrightarrow Y \leq F_X(x)).$$
- Тогда: $P(X \leq x) = P(F_X^{-1}(Y) \leq x) = P(Y \leq F_X(x)) = F_{Y \sim U(0,1)}(F_X(x)) = F_X(x)$
для $F_X(x) \in [0, 1]$ и $\forall x \in \mathbb{R}$. Теорема доказана.

- Теорема говорит о том, что если у нас есть случайная величина X , и задано преобразование $f: X \rightarrow Y$, имеющее все свойства функции распределения, то для того, чтобы величина Y была равномерно распределена на $(0,1)$, величина X должна быть распределена по закону, задаваемому функцией распределения $F_X = f$.
- Значит, если $Y \sim \mathcal{U}(0,1)$, то $X = F_X^{-1}(Y) \sim F_X$.
- Поэтому, чтобы получить семплы из распределения F_X , нужно:
 1. Получить семплы $y_i \sim \mathcal{U}(0,1)$. Это можно сделать с помощью функции генерации псевдослучайных чисел (**rand()** или **random()**).
 2. Выполнить преобразование $x_i = F_X^{-1}(y_i)$. Получим $x_i \sim F_X(x)$.
- **Для того, чтобы воспользоваться алгоритмом, нужно знать F_X и F_X^{-1} !**

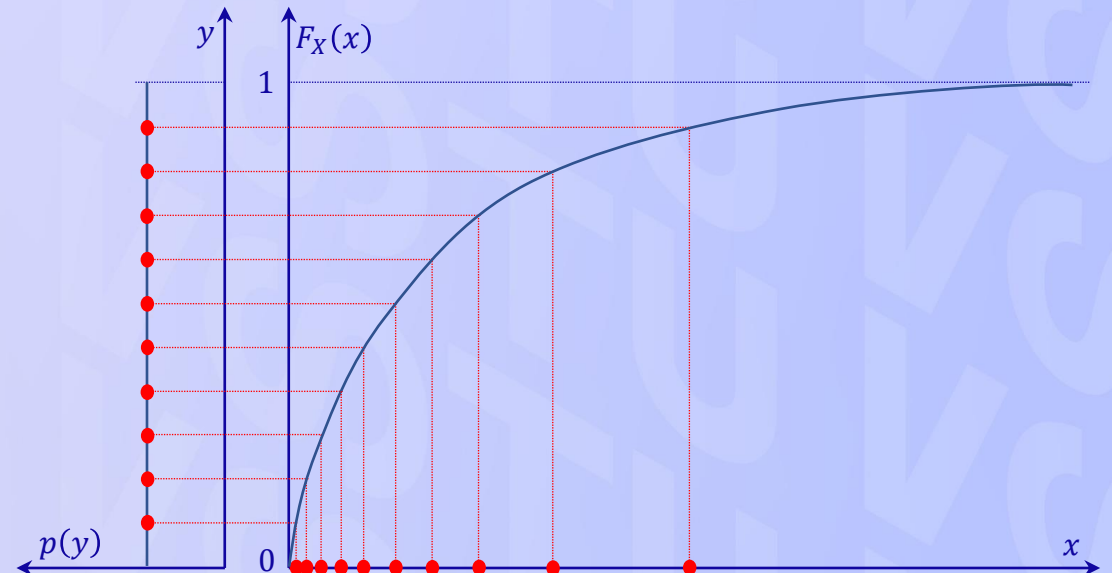
Пример с использованием теоремы об обратной функции распределения

- Рассмотрим экспоненциальное распределение: $F_X(x) = 1 - e^{-x}$, $x \geq 0$.
- Тогда $F_X^{-1}(y) = -\ln(1 - y)$, $y \in [0, 1)$.
- Действительно, $F_X^{-1}(F_X(x)) = -\ln(1 - (1 - e^{-x})) = -\ln(e^{-x}) = x$.
- Следовательно, если $Y \sim \mathcal{U}(0, 1)$, то:
$$X = F_X^{-1}(Y) = -\ln(1 - Y).$$

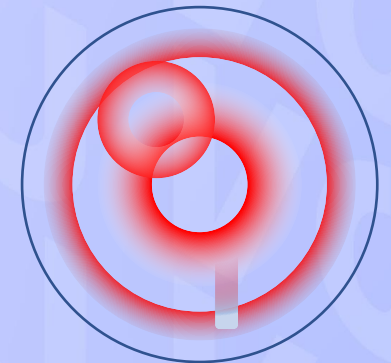
- Проверим:

$$\begin{aligned} P(X \leq x) &= P(-\ln(1 - Y) \leq x) = \\ &= P(1 - Y \geq e^{-x}) = P(Y \leq 1 - e^{-x}) = \\ &= F_{Y \sim \mathcal{U}(0,1)}(1 - e^{-x}) = 1 - e^{-x} = F_X(x). \end{aligned}$$

- Видно, что случайная величина X имеет распределение F_X .

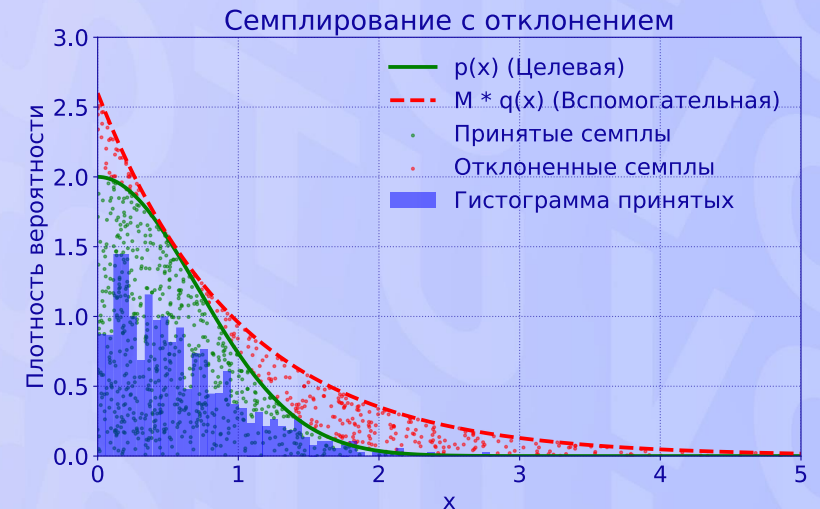
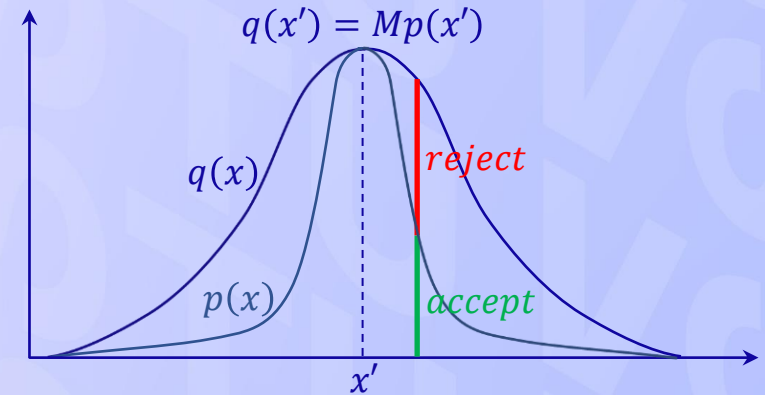


- Семплирование с помощью *теоремы об обратной функции* возможно только для относительно простых распределений.
- Рассмотрим следующий пример. Вы играете в игру: бросаете дротики в мишень. Но мишень особая – участки, соответствующие наибольшему числу очков, сложные и выделены цветом. Попасть в них очень тяжело.
- Чтобы облегчить себе задачу, Вы изготавливаете специальный трафарет и накладываете его на мишень. Трафарет сделан таким образом, что его толщина разная: на участках, соответствующих ярким областям мишени, она очень тонкая, а на бледных – наоборот, толстая.
- Чем толще участок трафарета, тем меньше шансов у дротика пробить его.



- Пусть дана случайная величина X с плотностью вероятности $p(x)$, для которой требуется получить семплы.
- Выберем вспомогательную случайную величину Q (из которой легко семплировать) с плотностью вероятности $q(x)$, $\text{supp}(p(x)) \subseteq \text{supp}(q(x))$, и число $M \in \mathbb{R} : M \geq \sup_x \frac{p(x)}{q(x)}$.
- For $k = 1..K$:
 - Получим семпл $y \sim q(x)$;
 - Получим семпл $u \sim \mathcal{U}(0, 1)$;
 - Если $u < \frac{p(y)}{Mq(y)}$, то принимаем y как семпл из X .

Семплирование с отклонением (Rejection Sampling)



- Таким образом, идея метода семплирования с отклонением заключается в том, что вместо того, чтобы генерировать семпл непосредственно из X , можно его получать из вспомогательного (*proposal*) распределения Y и принимать с вероятностью $\frac{p(x)}{Mq(x)}$, повторяя семплирование до тех пор, пока значение не будет принято.
- Покажем корректность метода. Нужно доказать, что *распределение принятых семплов x соответствует целевому распределению $p(x)$* .
 1. Рассмотрим *вероятность* того, что на некотором шаге алгоритма семпл $x \sim q(x)$ будет принят.

Условие принятия: $u \leq \frac{p(x)}{M \cdot q(x)}$, где $u \sim \mathcal{U}(0, 1)$.

Тогда вероятность принятия для некоторого фиксированного x :

$$P(\text{accept} | x) = P\left(u \leq \frac{p(x)}{M \cdot q(x)}\right) = F_{U \sim U(0,1)}\left(\frac{p(x)}{M \cdot q(x)}\right) = \frac{p(x)}{M \cdot q(x)},$$

причём $0 \leq \frac{p(x)}{M \cdot q(x)} \leq 1$ по условию алгоритма.

В этом случае, общая вероятность принятия семпла (усреднённая по всем $x \sim q(x)$) равна:

$$P(\text{accept}) = \int_{-\infty}^{+\infty} P(\text{accept} | x) \cdot q(x) dx = \int_{-\infty}^{+\infty} \frac{p(x)}{M \cdot q(x)} \cdot q(x) dx = \frac{1}{M} \int_{-\infty}^{+\infty} p(x) dx = \frac{1}{M}.$$

2. Найдём распределение принятых семплов $p_{acc}(x)$. По теореме Байеса:

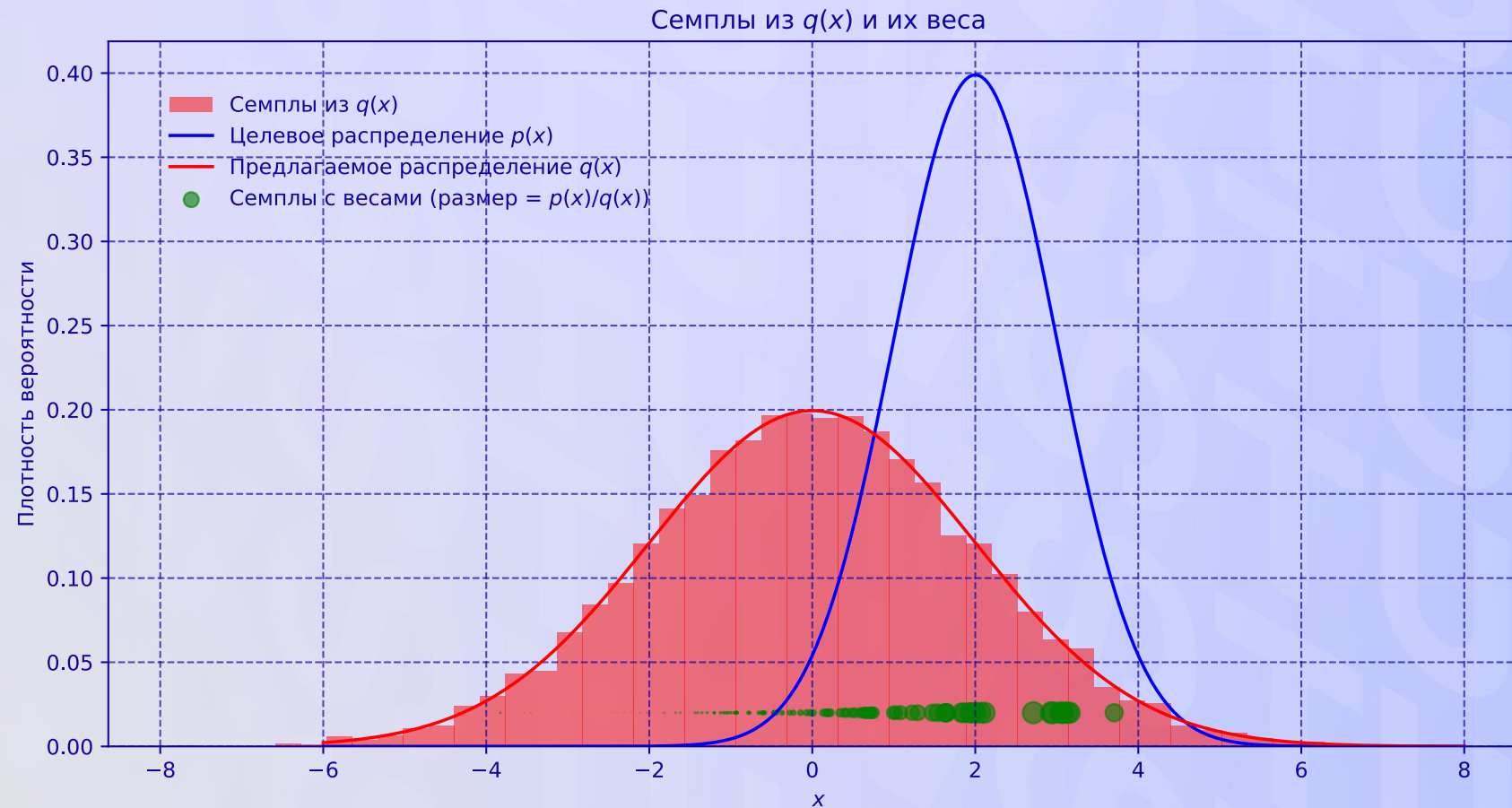
$$P(x | \text{accept}) = \frac{P(\text{accept} | x) \cdot q(x)}{P(\text{accept})} = \frac{p(x)/M}{1/M} = p(x) \Rightarrow \boxed{p_{acc}(x) = p(x)}.$$

- **Следует обратить внимание, что:**
 - ✓ Возможно семплирование из широкого класса распределений $p(x)$, если найти подходящее вспомогательное распределение $q(x)$. Однако, в многомерных пространствах вероятность принятия семплов может быть крайне низкой.
 - ✓ Метод подходит для случаев, когда целевое распределение известно с точностью до нормирующей константы.
 - ✓ Эффективность метода сильно зависит от выбора вспомогательного распределения $q(x)$ и константы M . Если M выбрана неудачно, то число отклонений будет большим, а сам метод – вычислительно затратным.
 - ✓ Для удачного подбора M нужно иметь представление о форме $p(x)$.
 - ✓ Метод часто используется в качестве вспомогательного.

- Рассмотрим способ оценки матожидания, не требующий семплов из целевого распределения. Введем вспомогательное распределение $q(x)$, $\text{supp}(p(x)) \subseteq \text{supp}(q(x))$, и запишем эстиматор как функцию семплов из него:

$$\mathbb{E}[f(x)] = \int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)} q(x)dx \approx \frac{1}{N} \sum_{i=1}^N f(x_i) \frac{p(x_i)}{q(x_i)}, \quad x_i \sim q(x).$$

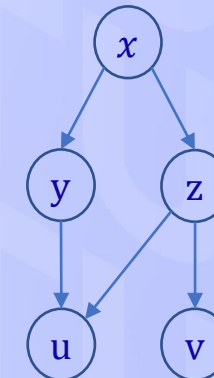
- Таким образом, $\mathbb{E}[f(x)]$ рассчитывается как матожидание по $q(x)$ от функции, домноженной на отношение $\frac{p(x_i)}{q(x_i)}$, называемое *likelihood ratio*.
- Для успешного семплирования также нужно правильно подобрать плотность распределения $q(x)$.
- Данный метод характеризуется высокой дисперсией за счет наличия областей, в которых $p(x) \gg q(x)$ (если $q(x)$ плохо приближает $p(x)$).



- Рассмотрим совместное распределение случайных величин: $p(x_1, x_2, \dots, x_n)$ представленное в виде графической модели.
- Как семплировать из такого распределения?
- Выполняем семплирование величин, находящихся в корневых вершинах.
- Затем *продвигаемся по дереву* до листовых вершин, выполняя семплирование для всех случайных величин с учетом полученных ранее семплов в выражениях условной вероятности.

- Например: $p(x, y, z, u, v) = p(x)p(y | x)p(z | x)p(u | y, z)p(v | z);$

$$\begin{aligned}x^* &\sim \mathcal{U}(0, 1); \\ y^* &\sim \mathcal{N}(x^*, 1); \\ z^* &\sim \mathcal{N}(0, 2x^*); \\ u^* &\sim \mathcal{U}(y^*, z^*); \\ v^* &\sim \mathcal{N}(z^*, 10).\end{aligned}$$



- Для выполнения байесовского моделирования используются **фреймворки вероятностного программирования (PPL)**, позволяющие объединить возможности современных языков программирования с эффективным расчётом параметров случайных величин.
- Одним из таких языков является **Pyro**, построенный на основе Python и PyTorch.



- Вероятностная модель в Pyro определяется функциями-примитивами, поведение которых может изменяться *в зависимости от контекста*.
- Модель создается как функция, в которой определяются функции-примитивы и зависимости между ними.

Элементы вероятностной модели	Функции-примитивы
Латентная случайная переменная	sample (<u>name</u> : str, <u>fn</u> : TorchDistributionMixin, * <u>args</u> , <u>obs</u> : Optional[torch.Tensor] = None , ...)
Наблюдаемая случайная переменная	sample (<u>name</u> : str, <u>fn</u> : TorchDistributionMixin, * <u>args</u> , <u>obs</u> : Optional[torch.Tensor] = torch.tensor([...]) , ...)
Обучаемые параметры	param (<u>name</u> : str, <u>init_tensor</u> : Optional[Union[torch.Tensor, Callable[[], torch.Tensor]]] = None, <u>constraint</u> : torch.distributions.constraints.Constraint = Real(), ...)
«Планки»	plate (<u>name</u> : str, size: Optional[int] = None, ...)

- *Функции* в Pyro выполняют свою основную функцию и дополнительно реализуют сторонние эффекты, причем поведение этих эффектов и основной функции может меняться в зависимости от контекста выполнения.
- Данный *контекст* в Pyro формируется с помощью «**трасс**» (traces).
- Трассы являются некоторой реализацией логов, в которых можно найти всю интересующую информацию о состоянии обрабатываемых объектов в Pyro и изменениях, которые произошли с ними после вызова функций, а также о том, как контекст выполнения повлиял на эти изменения.
- Для записи трассы функцию необходимо обернуть в:
traced_fn = routine.trace(**fn**)

- Вызов `traced_fn.get_trace(*args, **kwargs)` позволяет *выполнить функцию **fn**, сформировать трассу и вернуть объект, её представляющий.*

Пример вызова трассы	Пример трассы
<pre>def sample_fn(): return pyro.sample("x", pyro.distributions.Normal(0, 1)) # Обернем функцию в трассу traced_fn = poutine.trace(sample_fn) # Выполнение функции и получение трассы trace = traced_fn.get_trace() # Вывод информации о событиях в трассе print(trace.nodes)</pre>	<pre>{ 'x': { # Имя узла 'type': 'sample', # Тип узла (семпл) 'name': 'x', # Имя семпла 'is_observed': False, # Не наблюдаемая 'fn': Normal(loc: 0.0, scale: 1.0), 'value': tensor(0.1234), 'scale': 1.0, 'log_prob': tensor(-0.9189), 'args': (), # Аргументы вызова 'kwargs': {} # Аргументы вызова } }</pre>

- **Pyro** — это библиотека компонуемых обработчиков (handlers) сторонних эффектов для определения и изменения поведения программ.
- Её использование упрощает реализацию новых алгоритмов байесовского инференса.
- Обработчики могут использоваться как функции, декораторы, менеджеры контекста для изменения поведения функций или блоков кода.
- Примерами обработчиков являются:
 - ✓ *block*
 - ✓ *condition*
 - ✓ *trace*
- Пример реализации функциональности Pyro: Minipyro.

Пример реализации модели линейной регрессии в Pyro

```
def model(x, y=None): #  $x \in \mathbb{R}^{n \times m}$ ,  $y \in \mathbb{R}^n$ 
    sigma = 1.0
    sigma_0 = 3.0

    bias = pyro.sample("bias",
                        dist.Normal(0.0, sigma_0))

    with pyro.plate("features", x.shape[1]):
        weights = pyro.sample(
            "weights", dist.Normal(0.0, sigma_0))

    mean = x @ weights + bias

    with pyro.plate("data", x.shape[0]):
        return pyro.sample(
            "obs",
            dist.Normal(mean, sigma),
            obs=y)
```

Априорное распределение

Plate

Априорное распределение

Функция правдоподобия

Внутри `pyro.sample("name", distribution)` происходит следующее:

- ✓ Если это первый вызов `pyro.sample` с данном именем, генерируется случайная величина из указанного распределения (distribution).
- ✓ Значение данной случайной величины сохраняется в трассе Pyro (trace).
- ✓ Последующие вызовы `pyro.sample` с тем же именем не создают новое значение, а возвращают сохраненное. Это обеспечивает *согласованность* значений величин в рамках прохода модели.

- ✓ При вызове `trace.compute_log_prob()` будут рассчитаны `log_prob` всех величин в трассе, а также общая сумма `log_prob`.
- ✓ Если указан аргумент "obs", то вычисляется `log_prob` наблюдаемых данных, а не семплированных.
- ✓ Суммарный `log_prob` представляет собой логарифм совместной вероятности всех величин и данных.

Демонстрация практических примеров

Заключение

1. Вспомнили основные понятия классического машинного обучения.
2. Рассмотрели и сравнили классический и байесовский подходы к ML.
3. Определили MAE и MAP как частный случай байесовского подхода.
4. Сравнили частотный и байесовский подходы к описанию событий в ML.
5. Определили основную проблему байесовского подхода и выяснили способы её решения.
6. Разобрались, как применять методы Монте-Карло для оценки интегралов.
7. Дали понятие семплированию случайной величины, рассмотрели способы семплирования и их особенности применения.
8. Рассмотрели фреймворк Pyro и принципы его работы, на примерах разобрались с основными правилами работы с ним.

Спасибо за внимание!

Волгоград 2025
