4. Поиск оптимальной цены для одного товара (статическая, параметрическая постановка)

Постановка задачи

Формальная постановка задачи

Максимизировать прибыль $\boldsymbol{y(p, \xi)}$ по цене $\boldsymbol{p} \in [\boldsymbol{p}, +\infty)$.

4 разные постановки:

1. $p^* = \underset{p}{\operatorname{argmax}} E[y(p_n, \boldsymbol{\xi})]$ за n итераций.
2. $p^* = \underset{p}{\operatorname{argmax}} E[y(p, \boldsymbol{\xi})]$ за минимальное время.
3. $\max\limits_{p} \sum E[y(p, \boldsymbol{\xi})]$ Оптимизация траектории.
4. Найти зависимость $y(p)$

$E[\cdot]$ - может быть любым оператором. Можно считать, как мат. ожидание.

Решение за конечное число итераций

Параметрический подход

$\boldsymbol{y(p, \xi) = Q(p)(p - \underline{p})}$,

где $\boldsymbol{Q(p)}$ - число проданных товаров в зависимости от цены.

Линейная функция: $\boldsymbol{Q(p) = max(-ap + b, 0)}$
Гиперболическая функция: $\boldsymbol{Q\ (p) = max(-a/p + b, 0)}$
Экспоненциальная функция: $\boldsymbol{Q(p) = max(-exp(ap + b)c + d, 0)}$
Показательная функция $\boldsymbol{Q(p) = max(ba^p + d, 0)}$

Шум версия 1

$\boldsymbol{y(p,\xi) = Q(p)(p - \underline{p})}$,
где $\boldsymbol{Q(x)}$ - число проданных товаров в зависимости от цены.

Линейная функция: $\boldsymbol{Q(p) = max(-ap + b, 0)}$
Гиперболическая функция: $\boldsymbol{Q\ (p) = max(-a/p + b, 0)}$
Экспоненциальная функция: $\boldsymbol{Q(p) = max(-exp(ap + b)c + d, 0)}$
Показательная функция $\boldsymbol{Q(p) = max(ba^p + d, 0)}$

$\boldsymbol{\xi \sim N(0, \sigma)}$

В чем минусы?

Рассмотрим пример.
Линейная функция: $\boldsymbol{Q(p) = max(-ap + b + \xi, 0)}$
$\boldsymbol{\xi \sim N(0, \sigma^2)}$

Есть не нулевая вероятность получить $\boldsymbol{Q(p) > 0}$ для “больших” значений p. Под большими здесь подразумеваются такие, что $\boldsymbol{-ax + b < 0}$.
[См. Jupyter]

Как можно побороть?

Рассмотрим пример.

Линейная функция: $Q(\boldsymbol{p}) = \boldsymbol{max}(-\boldsymbol{ap} + \boldsymbol{b} + \boldsymbol{\xi}, \boldsymbol{0})$

$\boldsymbol{\xi} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma}(\boldsymbol{p}))$

$\boldsymbol{\sigma}(\boldsymbol{p})$ - убывает по p.

Или

$\boldsymbol{Q}(\boldsymbol{x}) = \boldsymbol{max}(-\boldsymbol{ap} + \boldsymbol{b} + \boldsymbol{\sigma}^{\boldsymbol{2}}(\boldsymbol{p})\boldsymbol{\xi}, \boldsymbol{0})$ и $\boldsymbol{\xi} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma}^{\boldsymbol{2}})$

Что выглядит не очень «академично». Но для ряда случаев приводит к «удобным» аналитическим решениям.

Шум версия 2

$\boldsymbol{y(p,\xi) = Q(p)\xi(p - \underline{p})}$,
где $\boldsymbol{Q(p)}$ - число проданных товаров в зависимости от цены.

Линейная функция: $\mathrm{Q}(\boldsymbol{p}) = \boldsymbol{max(-ap + b, 0)}$
Гиперболическая функция: $\boldsymbol{Q(x) = max(-a/p + b, 0)}$
Экспоненциальная функция: $\boldsymbol{Q(x) = max(-exp(ap + b)c + d, 0)}$

$\boldsymbol{E[\xi] = 1}$ (несмещенный шум)
$\boldsymbol{0 \leq \xi}$ – продажи не могу быть отрицательными

[См. Jupyter]

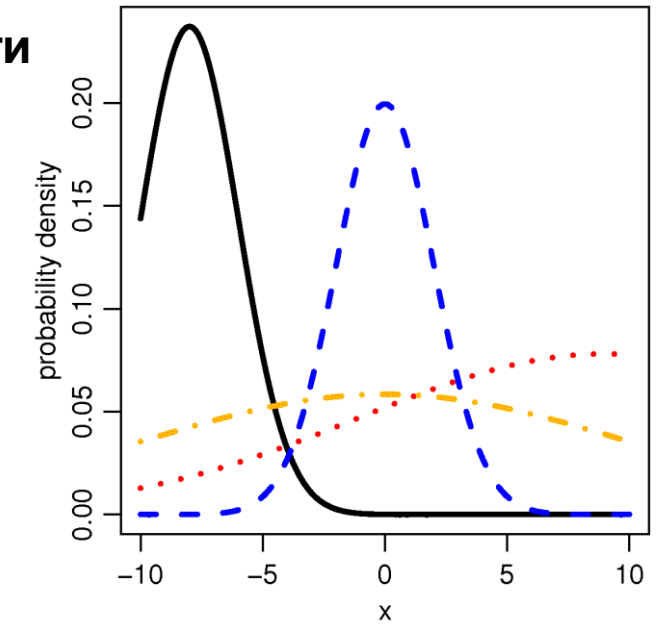
Какие распределения подойдут

1. Равномерное распределение
2. Нормальное распределение (т.к. на практике $\boldsymbol{\sigma^2} < 0.1$)
3. Усеченное нормальное распределение
   https://en.wikipedia.org/wiki/Truncated_normal_distribution
4. Треугольное распределение (распределение Симпсона)
   https://en.wikipedia.org/wiki/Triangular_distribution
5. Распределения симметричные относительно «1»

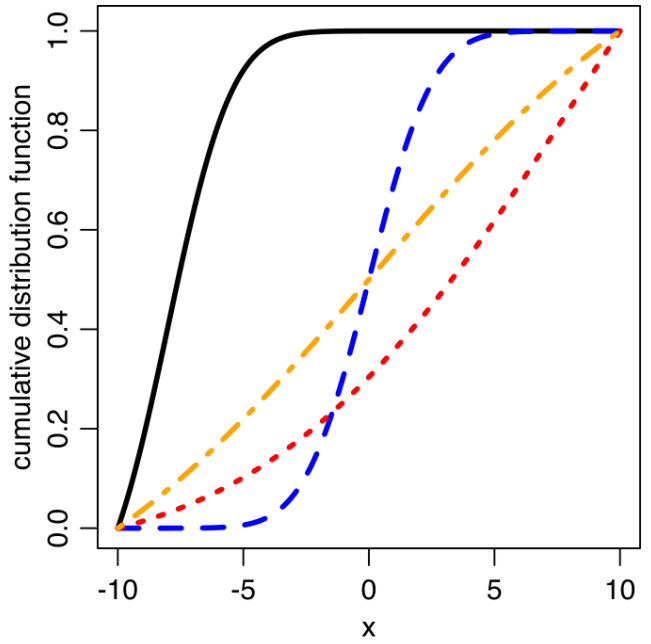
Усеченное нормальное распределение

Усеченное нормальное распределение — это распределение вероятностей, полученное из распределения нормально распределенной случайной величины путем ограничения случайной величины либо снизу, либо сверху (или обоих).

Функция плотности вероятности



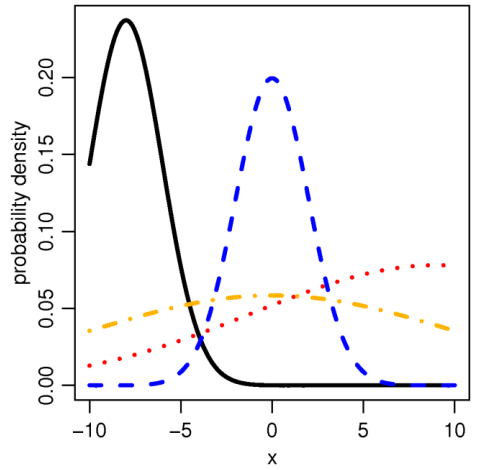
Кумулятивная функция распределения



Усеченное нормальное распределение

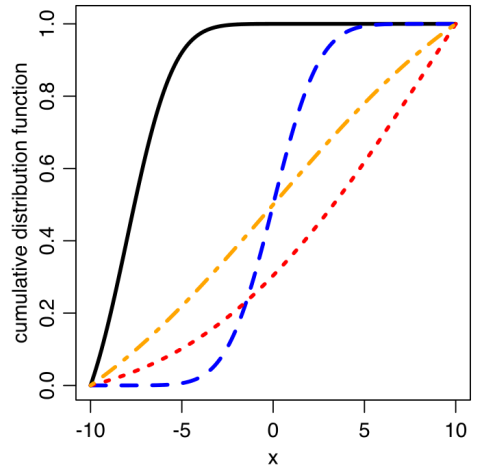
Функция плотности вероятности

$$f(x;\mu,\sigma,a,b) = \begin{cases} \frac{1}{\sigma}\frac{\phi(\frac{x-\mu}{\sigma})}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi(\frac{a-\mu}{\sigma})}, a \le x \le b \\ 0, \quad x \in [a,b] \end{cases}$$



Кумулятивная функция распределения

$$\phi(\xi) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\xi^2\right)$$



Давайте сравним «шумы»

[См. Jupyter]

Система координат

Вообще говоря, неудобно работать в пространстве (цена, прибыль).
Удобнее работать в пространстве (процентная наценка, прибыль).
Т.е. сделаем такую замену: $\frac{p-\underline{p}}{\underline{p}} = \boldsymbol{margin}\ (\boldsymbol{m})$

$$\boldsymbol{y(p,\xi) = Q(p)(p - \underline{p})},$$

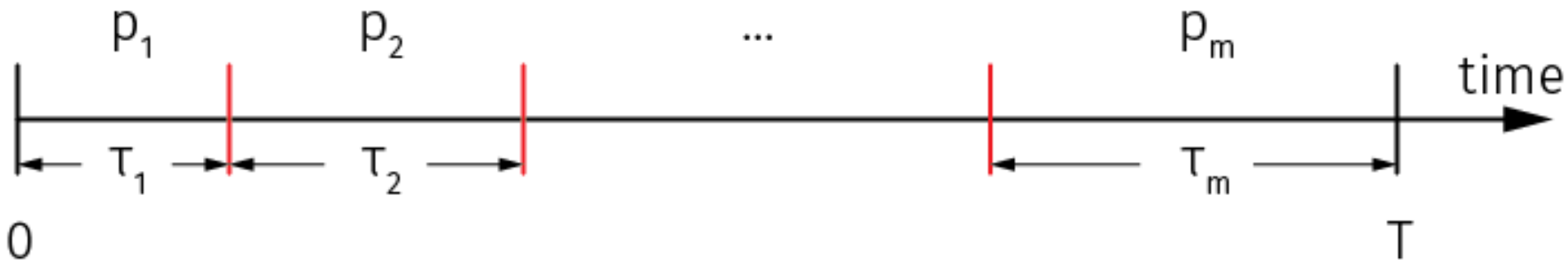
Можно переписать в виде:

$$\boldsymbol{y(p,\xi) = \xi \cdot \underline{p} \cdot Q(m)m},\ \boldsymbol{m \in [0, \bar{m})}$$

Линейная функция: $\mathrm{Q}(\boldsymbol{m}) = \boldsymbol{max(-am + b, 0)}$
Гиперболическая функция: $\boldsymbol{Q(x) = max(-a/m + b, 0)}$
Экспоненциальная функция: $\boldsymbol{Q(x) = max(-\exp(am + b)c + d, 0)}$

Сценарии построения ценообразования

Сценарий, при котором спрос остается постоянным в течение жизненного цикла продукта, но количество изменений цен ограничено ценовой политикой продавца.



Предполагая, что полная продолжительность жизненного цикла продукта T известна продавцу заранее, цель состоит в последовательной оптимизации цен для временных интервалов (m), а также оптимизации продолжительности τ_i этих интервалов:

Поиск решения за n шагов

$$p^* = \underset{p}{\operatorname{argmax}}\, E[y(p_n, \boldsymbol{\xi})]$$

- **Математика:** Если n=1. $p^* = \underline{\boldsymbol{p}} + \text{"}\mathbf{1}\text{"}$ – минимальный шаг дискретизации.
- **На практике** (если есть другие товары). $p^* = \underline{\boldsymbol{p}} \cdot \boldsymbol{s}$, где s – средняя (оптимальная) наценка в категории (близких товарах).

Поиск решения за n шагов

$$m^* = \underset{m}{\operatorname{argmax}}\, E[y(m_n, \boldsymbol{\xi})]$$

Как оптимизировать, если:

- Если n=3?
- Если n=20?

Поиск решения за n шагов

$$m^* = \underset{m}{\operatorname{argmax}}\, E[y(m_n, \boldsymbol{\xi})]$$

Какой шаг мы будем делать последним? В точку, где мы прогнозируем, что будет максимальный оптимум «в среднем»

Поиск решения за n шагов

$$m^* = \underset{m}{\operatorname{argmax}} E[y(m_n, \boldsymbol{\xi})]$$

То есть на предпоследнем шаге мы имеем

$$\hat{y}(m \quad , \boldsymbol{\xi} | m_{n-1}, m_{n-2}, \dots m_1)$$

Случайная функция (т.е. есть плотность вероятности в каждой точки). Соотв. в каждой точке есть мат. ожидание.

$\hat{y}$ определена через ее параметры $\hat{\theta} = \hat{\theta}(m_{n-1}, m_{n-2}, \dots m_1)$.

$$m^* = \operatorname{argmax} E[\hat{y}(m \quad , \boldsymbol{\xi} | m_{n-1}, m_{n-2}, \dots m_1)]$$

Поиск решения за n шагов

$$m^* = \underset{m}{\operatorname{argmax}}\, E[y(m_n, \boldsymbol{\xi})]$$

На самом деле максимум можно посчитать заранее.

$\underset{m}{\operatorname{argmax}}\, \boldsymbol{y(m, \xi)} = \underset{m}{\operatorname{argmax}}\, E[\boldsymbol{\xi} \cdot \underline{\boldsymbol{p}} \cdot \boldsymbol{Q(m)m}] = \underset{m}{\operatorname{argmax}}\, \boldsymbol{Q(m)m}$

$\boldsymbol{(Q(m)m)' = Q'(m)m + Q(m) = 0}$ (если функция выпукла)

$\boldsymbol{m = -\frac{Q(m)}{Q'(m)}}$ (удобное представление)

Поиск решения за n шагов

$$m^* = \underset{m}{\operatorname{argmax}} E[y(m_n, \boldsymbol{\xi})]$$

$$m^* = \underset{m}{\operatorname{argmax}} \boldsymbol{Q}(\boldsymbol{m})\boldsymbol{m}$$

Если функция не выпукла – любой аналитический метод. Разбиваем на отрезки унимодальности. Ищем максимум для каждого как для выпуклой задачи. Ищем максимум из максимумов через параметры.

$$m^* = \mathrm{W}(\hat{\theta})$$

Таким образом, у нас есть (пусть и очень сложная) функция $\mathrm{W}(\hat{\theta})$. Мы можем оценить влияние каждого параметра в векторе $\hat{\theta}$ оптимум.

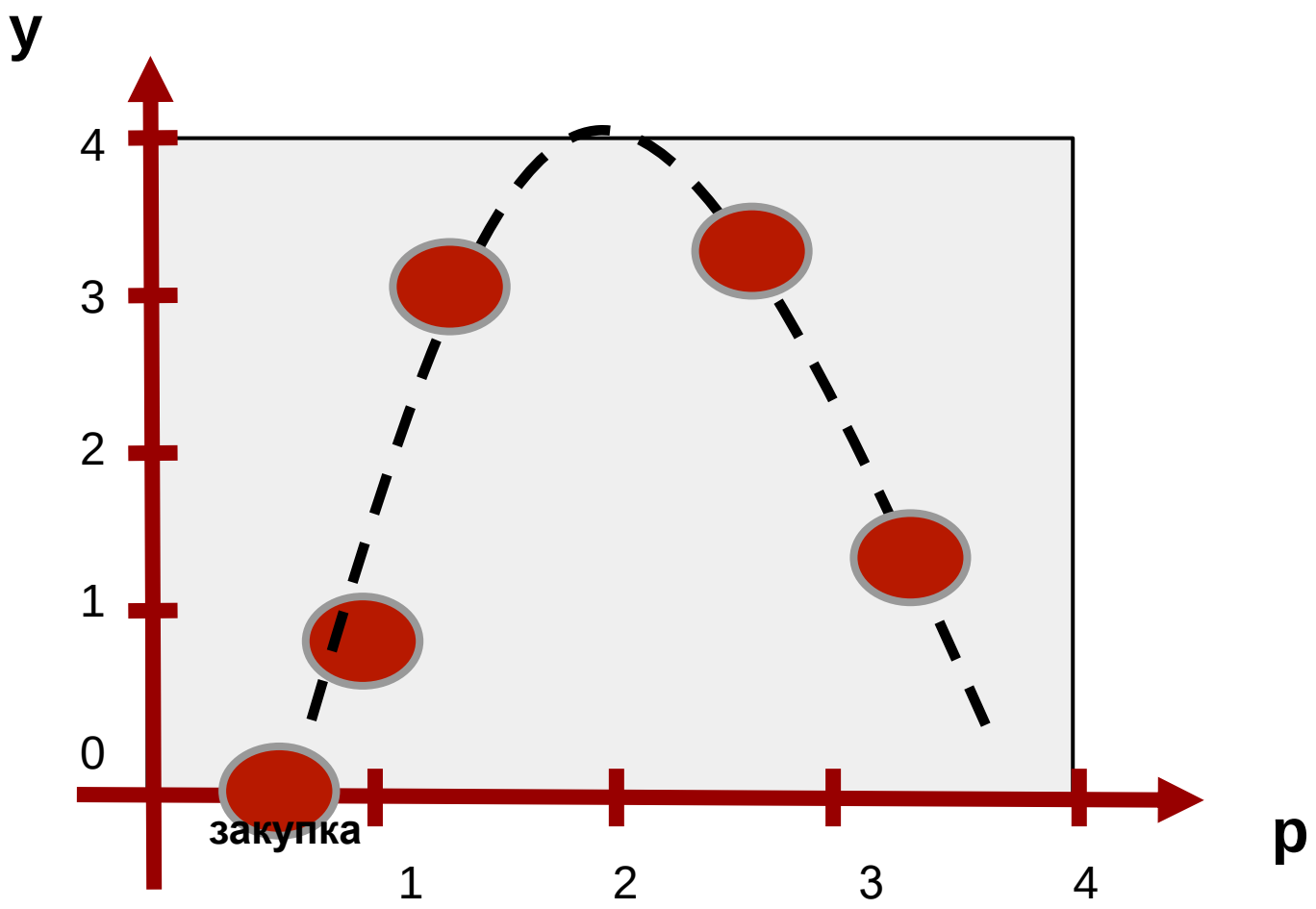
Поиск решения за n шагов

$$m^* = \mathrm{W}(\hat{\theta})$$

Таким образом, у нас есть (пусть и очень сложная) функция $\mathrm{W}(\hat{\theta})$.

Наша задача за n-1 шаг максимизировать качество аппроксимации $\mathrm{W}(\hat{\theta}_n(m_{n-1}|m_{n-2}, \dots m_1))$.
То есть выбрать такой m_{n-1}, что?

Нужно выбрать точку n-1



Поиск решения за n шагов

$$m^* = \mathrm{W}(\hat{\theta})$$

Таким образом у нас есть (пусть и очень сложная) функция $\mathrm{W}(\hat{\theta})$.

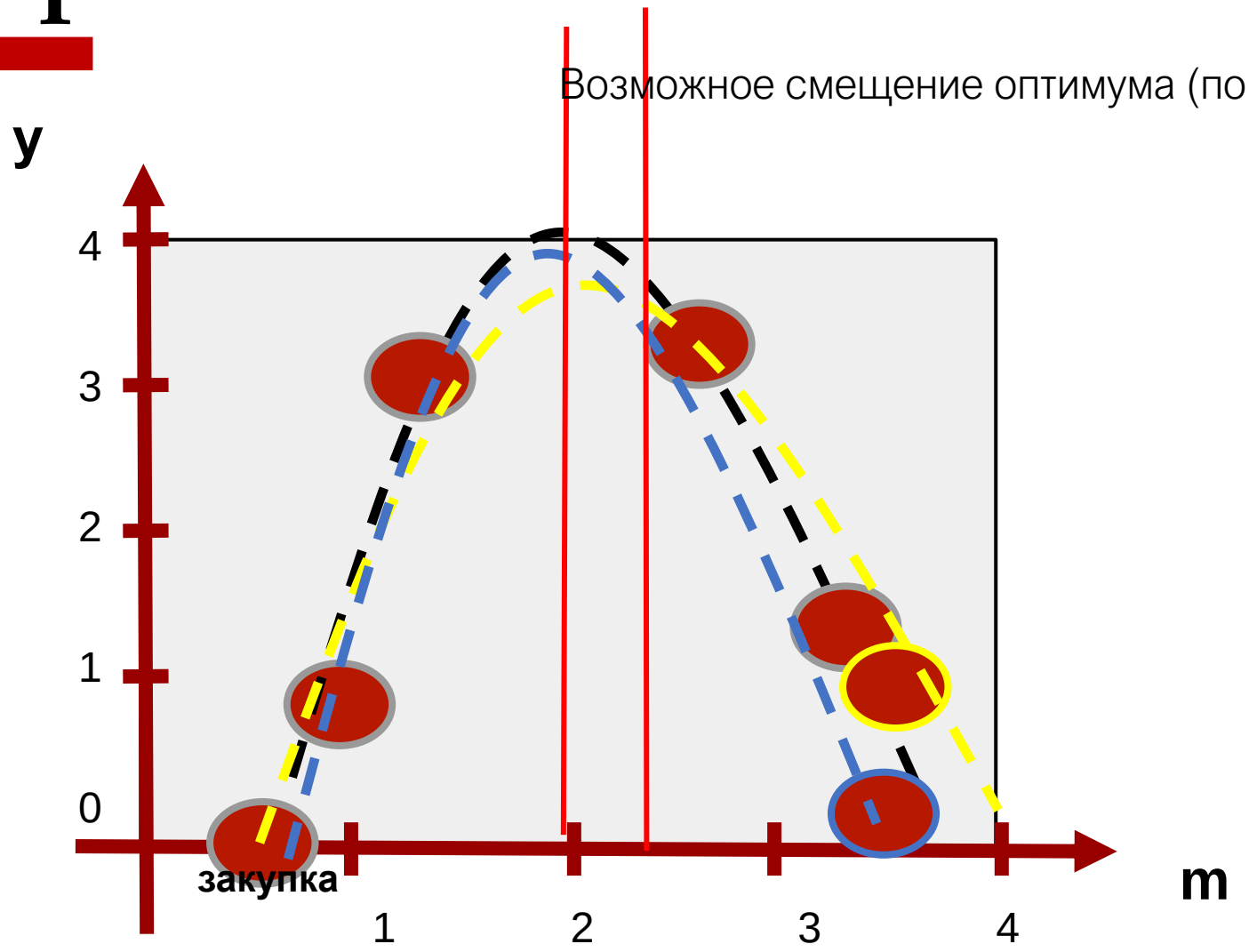
Наша задача за n-1 шаг максимизировать качество аппроксимации $\mathrm{W}(\hat{\theta}_n(m_{n-1}|m_{n-2}, \dots m_1))$.
То есть выбрать такой m_{n-1}, что?

Точность в предполагаемому АРГмаксимуме будет максимальной.
Это не минимум дисперсии! Это наилучшее уточнение по оси **m**.

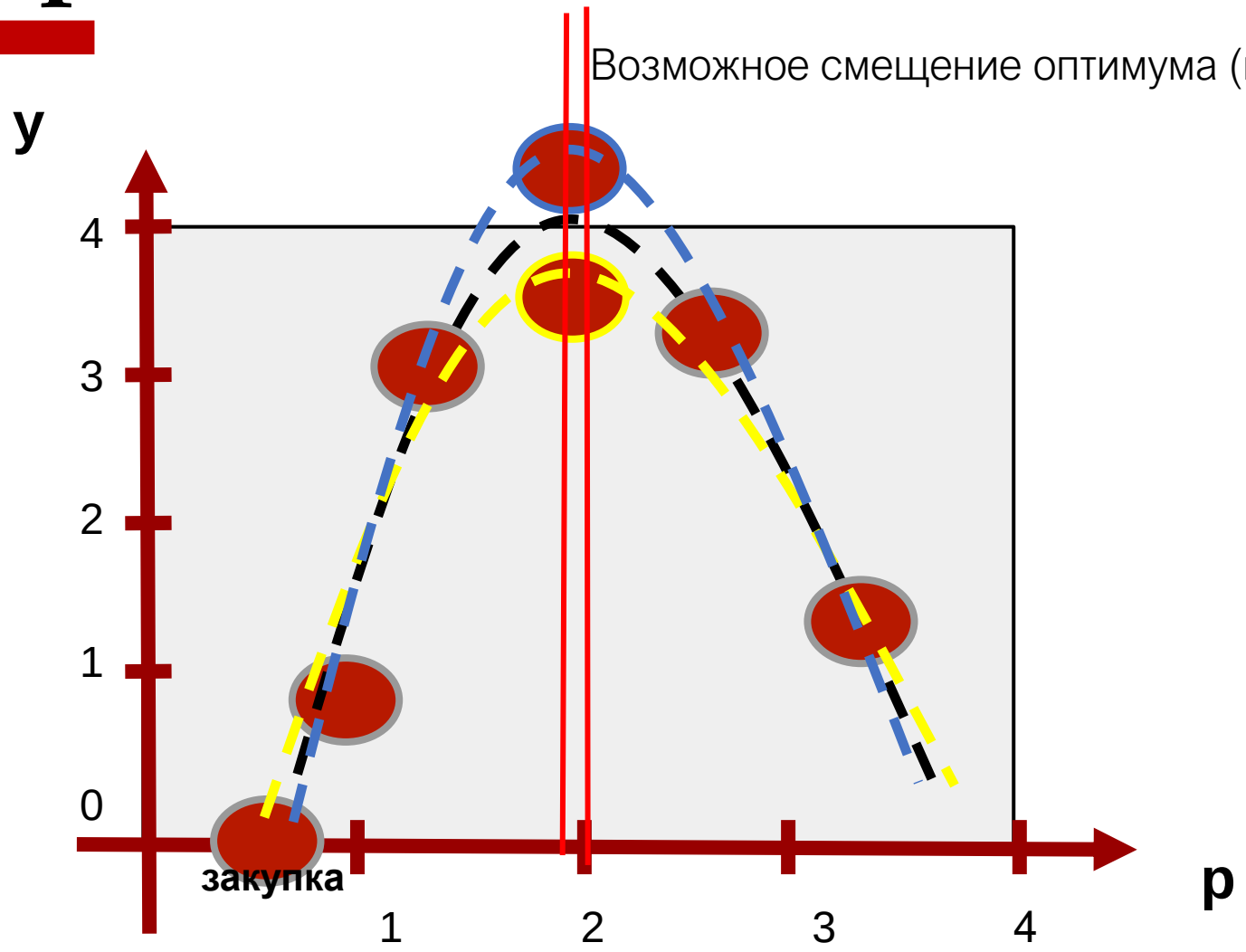
$$\max(W_n(\hat{\theta})) - \min(W_n(\hat{\theta})) \rightarrow \min$$

Вариант 1

Возможное смещение оптимума (по оси m).



Вариант 2



Поиск решения за n шагов

$$m^* = \mathrm{W}(\hat{\theta})$$

Таким образом у нас есть (пусть и очень сложная) функция $\mathrm{W}(\hat{\theta})$.

Наша задача за n-1 шаг максимизировать качество аппроксимации $\mathrm{W}(\hat{\theta}_n(m_{n-1}|m_{n-2}, \dots m_1))$.
То есть выбрать такой m_{n-1}, что?

Точность в предполагаемом АРГмаксимуме будет максимальной.
Это не минимум дисперсии! Это наилучшее уточнение по оси **m**.

Как правило, эти точки находятся на «краях» функции, не в оптимуме!

Поиск решения за n шагов

$$m^* = \mathrm{W}(\hat{\theta})$$

Какая у нас задача n-2 шаге?
Наша задача на n-2-ом шаге максимизировать качество аппроксимации $\mathrm{W}(\hat{\theta}_n(m_{n-2}|m_{n-3}, \dots m_1))$, так чтобы можно было его максимально улучшить на следующем.

Да, тут наступает метод Беллмана.

Если n не очень большое – можно сделать полный перебор.
Если n очень большое – считаем на допустимое n вперед.
Метод сокращения дисперсии – неплохая аппроксимация.

Метод сокращения дисперсии

Идея: выбирать объекты, которые максимально сократят дисперсию.

$$MSE(b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i, b))^2$$

$$MSE(\hat{b}) = \min_b(MSE(b))$$

$$D[MSE(\hat{b})] \approx \left(\frac{\partial f(x)}{\partial b}\right)^T \left(\frac{\partial MSE(b)}{\partial b^2}\right)^{-1} \left(\frac{\partial f(x)}{\partial b}\right)$$

$$x = arg\min_x D[MSE(\hat{b})]$$

Полный перебор

Допустим у нас *n* шагов, и *k* дискретизация по m.
На первом шаге у нас k альтернатив.
На втором шаге у нас k-1 альтернатива.
…
k(k-1)(k-2)…(k-n) – вариантов.
На практике k не больше 20. n не больше 10.
20!19!..11! = 670 442 572 800 (очень много)
10! = 3 628 800 (можно посчитать). Если перебирать 100 вариантов в секунду – это примерно 10 часов.

Можно пользоваться бинами

Создать иерархически бины.

1. Сначала разбить, например, на 4 бина.
2. Потом дискретизовать внутри оптимального бина.

	Цена от	Цена до	Ширина бина	Средняя
Большой бин 1	1000	1041	41	1020
	1041	1084	43	1062
	1084	1129	45	1106
	1129	1177	48	1153
	1177	1229	52	1203
Большой бин 2	1229	1285	56	1257
	1285	1346	61	1315
	1346	1413	67	1379
	1413	1487	74	1450
Большой бин 3	1487	1570	83	1528
	1570	1663	93	1616
	1663	1769	106	1716
Большой бин 4	1769	1891	122	1830
	1891	2034	143	1962
	2034	2203	169	2118

Поиск решения за n шагов

Плюсы постановки:

- Понятная постановка
- Для малого n – имеет право на существование.

Минусы постановки:

- Оптимальное решение будет часто тестировать функцию на «краях», то есть заведомо не в оптимуме.

2. За минимальное число шагов

$$m^* = \underset{m}{\operatorname{argmax}} E[y(m_n, \boldsymbol{\xi})]$$

$$m^* = \underset{m}{\operatorname{argmax}} \boldsymbol{Q}(\boldsymbol{m})\boldsymbol{m}$$

Что значит найти решение? $|m^* - m_n| \leq d$

Мы хотим построить такую последовательность по m, которая гарантирует, что мы отклонимся от оптимума не больше чем на d.

2. За минимальное число шагов

$$m^* = \underset{m}{\operatorname{argmax}} E[y(m_n, \boldsymbol{\xi})]$$

$$m^* = \underset{m}{\operatorname{argmax}} \boldsymbol{Q}(\boldsymbol{m})\boldsymbol{m}$$

Аналогичные рассуждения, что и для предыдущей задачи.

$$m^* = \mathrm{W}(\hat{\theta})$$

Таким образом, у нас есть (пусть и очень сложная) функция $\mathrm{W}(\hat{\theta})$.

Только теперь у нас есть критерий остановки в виде

$$\left|\max(m_n(\hat{\theta})) - \min(m_n(\hat{\theta}))\right| \leq d$$

Ну или $\left|\max(W_n(\hat{\theta})) - \min(W_n(\hat{\theta}))\right| \to \min$ (критерий оптимизации)

Метод Беллмана.

За минимальное число шагов

Плюсы постановки:

- Удобная постановка

Минусы постановки:

- Оптимальное решение будет часто тестировать функцию на «краях» то есть заведомо не в оптимуме.

3. Оптимизации траектории

$$m^* = \operatorname{argmax}_m E[y(m_n, \boldsymbol{\xi})]$$

$$m^* = \operatorname{argmax}_m \boldsymbol{Q}(\boldsymbol{m})\boldsymbol{m}$$

В целом по-прежнему можно получить такую функцию:

$$m^* = \mathrm{W}(\hat{\theta})$$

Но только теперь у нас другая задача:

$$\max_m \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

$\boldsymbol{Q_i}$ - оценка функции $\boldsymbol{Q}$ до i-ого шага.

3. Оптимизации траектории

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Если число шагов бесконечно? То в целом не очень важно как искать! :)

Считаем, что число шагов конечное:

1. Достаточно маленькое
2. Достаточно большое

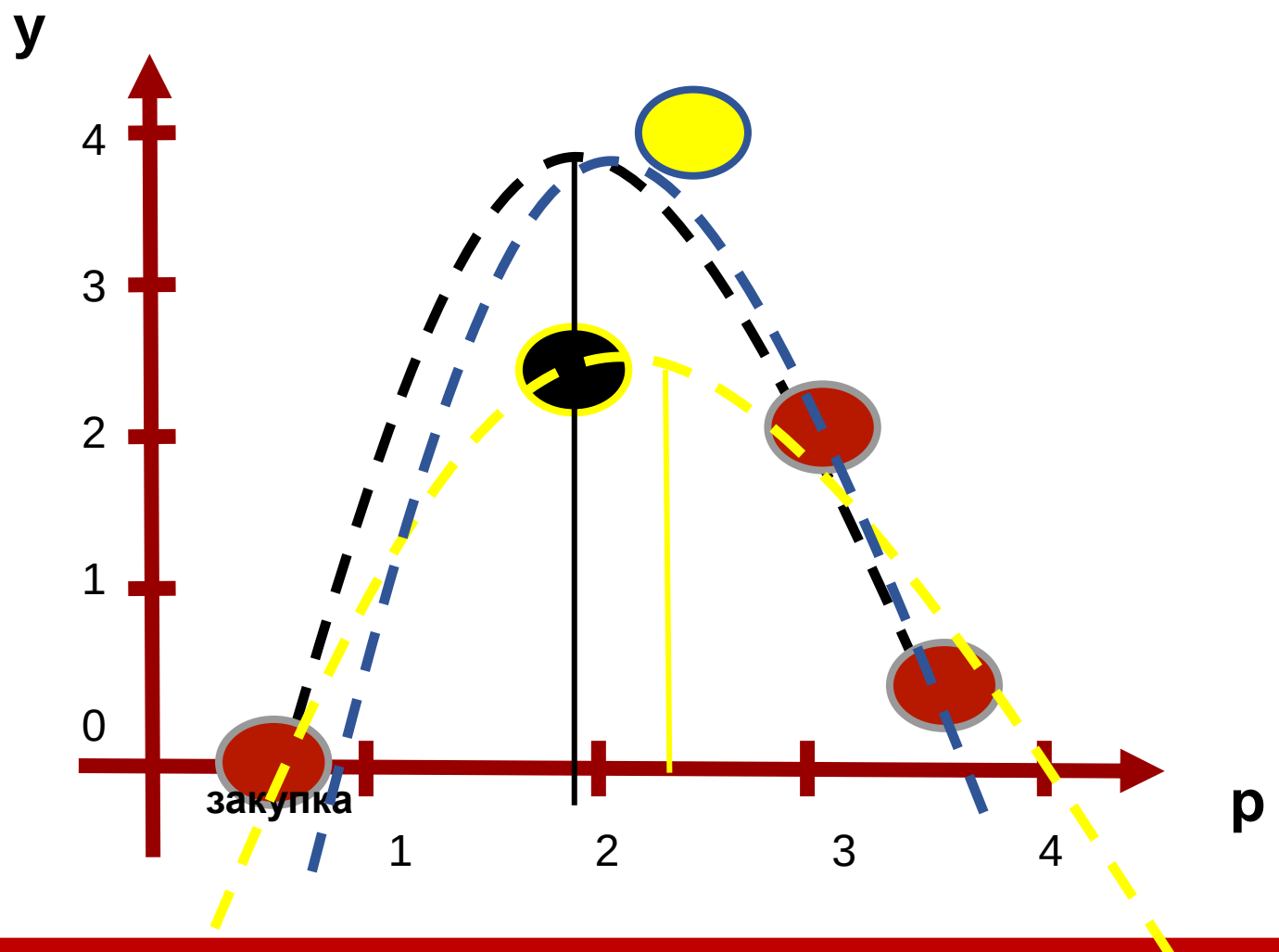
Наивный алгоритм

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Будем просто искать оптимум на каждом шаге и тестировать его.

$$m_1 = \operatorname*{argmax}_{m} \boldsymbol{Q_0}(\boldsymbol{m})\boldsymbol{m}$$
$$m_2 = \operatorname*{argmax}_{m} \boldsymbol{Q_1}(\boldsymbol{m})\boldsymbol{m}$$
$$\ldots$$
$$m_i = \operatorname*{argmax}_{m} \boldsymbol{Q_{i-1}}(\boldsymbol{m})\boldsymbol{m}$$

Последовательное тестирование точек



Плюсы и минусы наивного алгоритма

Плюсы:

- Простой
- На небольшом n хорошо работает
- Удобен на практике особенно если мы не очень хорошо параметризуем Q.

Минусы:

- Никак не учитывает число шагов
- Не старается улучшить аппроксимацию Q

3. Оптимизации траектории

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Если число шагов достаточно **маленькое**, то опять все тоже самое можно решить методом **Беллмана**. Ну или полный перебор ☺.
Только теперь у нас другой функционал качества.

3. Оптимизации траектории

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Если число шагов достаточно **большое**:

а) Считать Беллманом на несколько шагов вперед

б) «Бандиты» (будет отдельная лекция)

в) Решать exploration-exploitation tradeoff другими методами

Exploration-exploitation tradeoff

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Наш выбор состоит в следующем:
а) Выбирать оптимальную точку, но тогда мы будем (возможно) медленно искать оптимум
б) Выбирать не оптимальную точку, но тогда мы быстрее найдем точку.

Интуитивно понятно, что на первых шагах лучше бОльший приоритет отдавать exploration.

Exploration-exploitation tradeoff

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Наш выбор состоит в следующем:

а) Выбирать оптимальную точку, но тогда мы будем (возможно) медленно искать оптимум - $\mathrm{y_i} = \underset{m}{\mathrm{argmax}} \boldsymbol{Q_{i-1}}(\boldsymbol{m})\boldsymbol{m} = \mathrm{W_{i-1}}(\hat{\theta})$ $(\mathrm{y_a})$

б) Выбирать не оптимальную точку, но тогда мы быстрее найдем точку. Например на 1 шаг, это решение такой задачи.

$\max(W_{i-1}(\hat{\theta})) - \min(W_{i-1}(\hat{\theta})) \rightarrow \min$ Будем обозначать как $\mathrm{y_b}$

Exploration-exploitation tradeoff

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Здесь важно абсолютное значение максимума, уже не сам argmax.

$$\max(W_{i-1}(\hat{\theta})) - \min(W_{i-1}(\hat{\theta})) \rightarrow \min$$

Это сужение расстояния по m. Эта оптимизация явно ничего не говорит, о том как улучшиться в бедующем $\boldsymbol{Q_i}(m_{i+1})m_{i+1}$. Но этом можно оценить через размах значений **на этом интервале** для будущей функции.

$$\boldsymbol{\Delta y} = \underset{\boldsymbol{m} \in \boldsymbol{M}}{\mathrm{VAR}}[\boldsymbol{Q_i}(m_{i+1})m_{i+1}] = \max \boldsymbol{Q_i}(m_{i+1})m_{i+1} - \min \boldsymbol{Q_i}(m_{i+1})m_{i+1}$$

Exploration-exploitation tradeoff

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Таким образом, для каждой точки m, мы можем получить пару:

- $\operatorname{argmax}_{m} \boldsymbol{Q_{i-1}}(m_i)m_i$ - оценку в максимуме (для данного состояния)
- $\boldsymbol{Q_{i-1}}(m_i)m_i$ - оценку в значении
- $\boldsymbol{\Delta y}$ оценку уточнения оптимума по y (в лучшем случае).

Примечание: Можно использовать $\boldsymbol{\Delta y/2}$ или честно посчитать среднее.

Exploration-exploitation tradeoff

$$\max_{m} \sum E[y(m, \boldsymbol{\xi})] = \max_{m_1, m_2, \ldots} (\boldsymbol{Q_0}(m_1)m_1 + \boldsymbol{Q_1}(m_2)m_2 + \cdots)$$

Для каждого шага можно ввести такой функционал качества:

$$\boldsymbol{Q_{i-1}}(m_i)m_i + (\boldsymbol{n-1})\boldsymbol{\Delta y/2}$$

Мы складываем наш потенциальный результат с потенциальным возможным улучшением на оставшихся n-шагах.
Соответственно выбираем точку, которая максимизирует данный функционал.

Exploration-exploitation tradeoff

Мы рассмотрели одношаговую оптимизацию. Аналогично можно считать на 2 шага вперед.

Также существует много эвристик:

$$\underset{m}{\operatorname{argmax}} \boldsymbol{Q_{i-1}}(m_i)m_i + (\boldsymbol{n-1})\boldsymbol{\Delta y/10}$$

$$\underset{m}{\operatorname{argmax}} \boldsymbol{Q_{i-1}}(m_i)m_i + (\boldsymbol{n-1})^{\boldsymbol{0.8}}\boldsymbol{\Delta y/2}$$

Потому что добавка качества в долгую начинает терять в «весе».

Exploration-exploitation tradeoff

Дополнительно (на практике) применяют «штрафы» за удаление от известных точек. Пусть уже есть k наблюдений.

Возможные варианты:

- Ширина доверительного интервала.
- По одному удаляем уже известные прецеденты. Строим $\widetilde{y_j^k}\ (m_i)$ для k-го удаленного элемента. Вычисляем $\max_k \widetilde{y_j^k}\ (m_i) - \min_k \widetilde{r_j^k}\ (m_i)$.
- Можно отдельно штрафовать за отдаление $\min_k |m_i - m_k|$

Плюсы и минусы EE-алгоритма

Плюсы:

- Гарантированно не хуже наивного.

Минусы:

- Сложный
- На практике потребуется много эвристик, чтобы работал хорошо

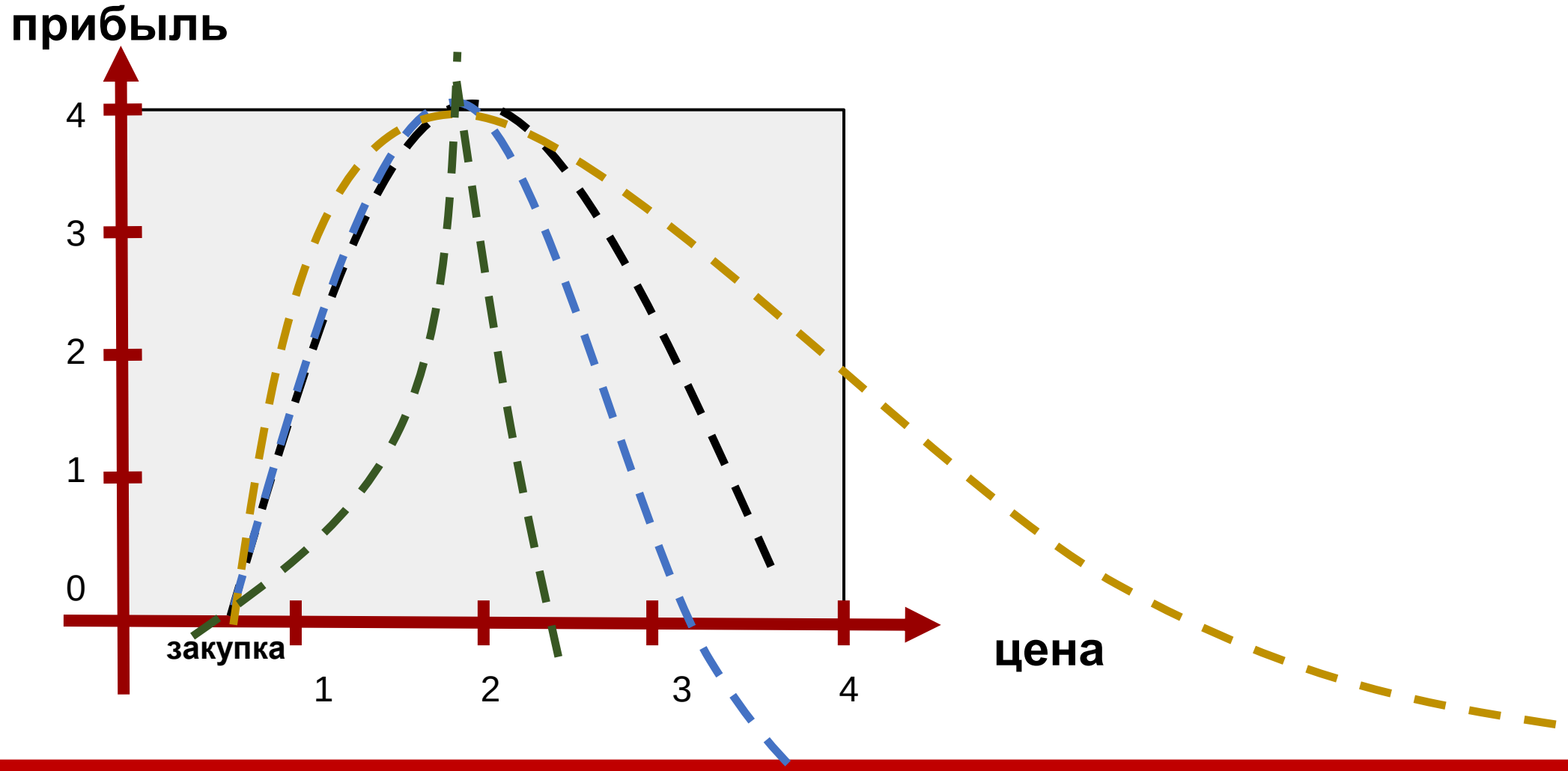
Обратите внимание: нам в целом было не важна насколько хорошо параметризованная Q, лишь бы она верно указывала на оптимум.

Пример

1. То есть часто может оказаться так, что, параметризация не самая лучшая, но она хорошо находит оптимум.
2. Много разных параметризаций находят один и тот же оптимум.

$$\underset{x}{\operatorname{argmax}}\, \boldsymbol{ax^2} = \underset{x}{\operatorname{argmax}}\, \boldsymbol{ax^4} = \underset{x}{\operatorname{argmax}}\, \boldsymbol{asin(x)}$$

Разные параметризации



4. Найти зависимость

$$Q^* = \boldsymbol{Q}(\boldsymbol{m})$$

Нужно найти восстановить зависимость *оптимально в некотором смысле*.

В некотором смысле классическая задача Active learning. См. соотв. лекцию.

На практике, как правило, есть более конкретная формализация.

4. Найти зависимость

$$Q^* = \boldsymbol{Q}(\boldsymbol{m})$$

Обычно интересно:

1) Хорошую аппроксимацию в районе оптимума.
2) Хорошая аппроксимацию на отрезке $\boldsymbol{m} \in [\boldsymbol{0}, \boldsymbol{m}^*]$

Формализуем 2-ую постановку.

4. Найти зависимость

Вариант 2.

$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Q_i - \hat{Q}\left(m_i\right)\right)^2$ для всех $m_i \in [\mathbf{0}, \boldsymbol{m}^*]$

Вариант 2. Учитываем вес ошибки: чем ближе к оптимуму, тем более значима ошибка.

$\mathrm{MSEW} = \frac{1}{nM}\sum_{i=1}^{n}\left(Q_i - \hat{Q}\left(m_i\right)\right)^2 m_i$ для всех $m_i \in [\mathbf{0}, \boldsymbol{m}^*]$

$\mathrm{M} = \sum m_i$ - нормировочная константа.

Плюсы и минусы постановки

Плюсы:

- Помимо оптимума находит еще и хорошую параметризацию
- Часто бывает нужна для планирования промо-акций, демпинга и др. активностей

Минусы:

- Если функция плохо параметризованна приводит к проблемам.

4. Семинар

Как на практике выглядит Q(p)