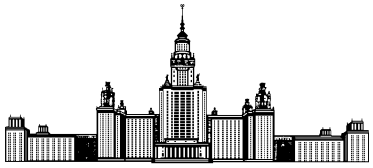
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

КУРСОВАЯ РАБОТА

Обзор методов оптимизации гиперпараметров
Review on hyperparameter optimization methods

Выполнил:

студент 3 курса 317 группы

Швец Павел Игоревич

Научный руководитель:

д.ф-м.н., профессор

Дьяконов Александр Геннадьевич

Москва, 2020

Содержание

Аннотация

В данной работе производится обзор существующих методов оптимизации гиперпараметров. Рассматриваются подходы, использующие переборные стратегии и стратегии, использующие байесовскую оптимизацию. В работе проведены сравнительные характеристики в зависимости от выбора вспомогательной функции и структуры вероятностной модели использующиеся при байесовской оптимизации.

1 Введение

Существует множество различных моделей используемых в машинном обучении. Однако почти все являются парметризуемыми (такие параметры называют гиперпараметрами), настройку которых предлагается осуществлять пользователю. На этот процесс уходит достаточно много времени. Поэтому возникает потребность в быстрых и эффективных методах их настройки.

Привычные градиентные методы оптимизации мало применимы в данной области, поскольку не всегда существует прямая функциональная зависимость между функцией качества (потерь) и гиперпараметрами. Далее приведён обзор существующих методов оптимизации гиперпараметров моделей. Приведены сравнительные эксперименты.

2 Постановка задачи

Пусть дано некоторое семейство $\mathcal{A}$ алгоритмов, параметризуемое N гиперпараметрами. Тогда будем обозначать Λ_k – пространство рассматриваемых параметров задаваемых для k-го гиперпараметра. Тогда общее пространство гиперпараметров алгоритма будем обозначать $\Lambda = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_N$. Будем обозначать $\mathcal{A}_\lambda$ алгоритм, параметризованный вектором гиперпараметров $\lambda \in \Lambda$.

Каждое из пространств Λ_k может быть действительным (например, параметры EM-алгоритма для смеси нормальных распределений или темп обучения для стохастического градиентного спуска), дискретным (например, число соседей в методе K – ближайших соседей), бинарным (например, использование или нет early stopping при обучении нейронной сети). Обычно подразумевается, что пространство ограничено (в случае задания с помощью множества конечно).

Также возможны ситуации, когда некоторые гиперпараметры не влияют на модель при не изменении других (например, если в модели параметризуется расстояние Минковского p, а рассматривается косинусное расстояние в методе K–ближайших соседей).

Пусть дана некоторая выборка $\mathcal{X}$, пусть также задана некоторый функционал качества $\mathcal{L}$. Тогда задачей является найти глобальный минимум следующего выражения:

$$\lambda_* = \arg\min_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{(\mathcal{X}_{train}, \mathcal{X}_{valid}) \sim \mathcal{X}} V(\mathcal{L}, \mathcal{A}_\lambda, \mathcal{X}_{train}, \mathcal{X}_{valid})$$

Здесь $\mathcal{X}_{train}, \mathcal{X}_{valid}$ – некоторое разделение выборки для обучения и валдиации. Под $V(\cdot, \cdot, \cdot, \cdot)$ подразумевается функционал в виде функции потерь, который показывает качетсво алгоритма $\mathcal{A}_\lambda$, обученного на $\mathcal{X}_{train}$ и проверяющийся на $\mathcal{X}_{valid}$.

Поскольку рассматриваются выборки конечного размера математическое ожидание приближается с помощью различных стратегий валидации (например, Hold-out, K-Fold, LOO, и т.д.), от выбора которой зависит затрачиваемое на подсчёт время.

Также иногда вводится параметр бюджета T, задающий ограничение на работу метода поиска параметра λ_*, примером тому может служить: максимальное количество итераций, максимальное время работы и т.д.

3 Обзор существующих методов

Ниже будут рассмотрены некоторые общие подходы и их модификации к решению поставленной задачи оптимизации гиперпараметров.

3.1 Grid Search

Один из самых используемых и простых методов при оптимизации гипрепараметров. Идея его заключается в переборе сетки задаваемых значений. Обычно данный метод используют при наличии соответствующих вычислительных ресурсов, поскольку его основным преимуществом является параллелизуемость. Ещё одним из преимуществ является применимость на всех структурных пространствах Λ. Поскольку сетка может содержать достаточно много узлов имеет место проклятие размерности. Пусть задано рассматриваемое структурное пространство Λ (в случае Grid Search оно дискретно), тогда число запусков обучения будет пропорционально зависеть от $|\Lambda| = \prod_{k=1}^{n} |\Lambda_k|$. Поэтому одновременно лучше настраивать небольшое количество параметров.

3.2 Random Serach

При использовании данного метода вводится параметр бюджета T, упомянутый ранее (в данном случае это количество итераций). Это надстройка над Grid Search, где априорные распределения на гиперпараметры заданы равномерным распределением $\sim U(\Lambda)$. В случае Random Search возможно задать априорное распределение на гиперпараметры с помощью любого другого распределения, подходящего под вид Λ_k. Например, если гиперпараметр бинарный, то можно взять параметризованное распределение Бернули $Bern(p)$, если же параметр может принимать любые действительные значения, то можно рассмотреть, например, нормальное распределение $\mathcal{N}(\alpha, \sigma^2)$, если только положительные, то гамма-распределение $\Gamma(\alpha, \beta)$. Также как Grid Search данный метод параллелизуем, однако не использует информацию о качестве работы алгоритма, происходит лишь сэмплирование гиперпараметров.

Помимо этого существуют вариации данного метода. Например, в работе [2] предложен метод Weighted Random Search (WRS). Его идея заключается в обновлении параметров для поиска в зависимости от их важности. Это происходит в две фазы:

1. Запуск N_0 итераций обычного Random Search. После которого происходит оценка важности признаков с помощью разложения fANOVA [4];

2. Пусть ранее была вычислена за k-шагов лучшая точка X^k максимизирующая критерий качества $F(\cdot)$. Также пусть w_i – соответствующая гиперпараметру i важность признака. Тогда сопоставим каждому признаку $p_i = w_i/w^*$ – вероятность изменения при поиске, где $w^* = \max\limits_i w_i$. Далее происходит модификация Random Search. С вероятностью p_i будет рассмотрена новая точка X^{k+1}, у которой координата i будет сэмплирована из заданного ранее априорного распределения и с вероятностью $1 - p_i$ изменена не будет.

Также в работе представлены теоретические обоснования о том, что вероятности достичь глобальный минимум помощью WRS выше, чем с помощью обычного RS.

3.3 Bayesian Optimisation

Байесовские методы оптимизации могут быть использованы как инструмент поиска глобального минимума функций, затраты на вычисления которых велики. Ос-

новная идея этого метода заключается в создании модели, которую можно обновлять и использовать при принятии решений об оптимизации. Поиск минимума происходит итеративно в две стадии: построение вероятностной модели и вспомогательной функции (acquisition function), отвечающей за выбор следующей точки поиска. Обычно в качестве априорного распределения вероятностной модели берётся гауссовский процесс, также известны варианты, использующие случайные леса или нейронные сети. Вспомогательная функция использует предсказанное распределение построенной вероятностной модели и решает проблему разведки и использования (exploration and exploitation trade-off).

3.3.1 Гауссовские процессы

В качестве вероятностных моделей можно использовать непараметрические. Примером такой является Гауссовский процесс [8]. Данная модель используется в тех случаях, когда параметрические модели становятся уже слишком простыми, чтобы описывать распределение, задаваемое вновь поступающими данными. Гауссовский процесс, конечно же, является параметризуемой моделью, поскольку априори задаётся класс рассматриваемых распределений (в данном случае нормальные). То есть также можно рассматривать процессы порождаемые другими распределениями, например, распределением Стьюдента [10]. Но так как класс распределений задан, то модель считается непраметризуемой и зависит лишь от выборки.

Опр. 1. *Гауссовским называется случайный процесс* $GP(\mu(\cdot), k(\cdot,\cdot))$*, каждое конечномерное распределение которого имеет вид нормального распределения. Данный случайный процесс задаётся своими функцией среднего* $\mu(\cdot)$ *и ковариационной функцией* $k(\cdot,\cdot)$*.*

Для задачи оптимизации гиперпараметров удобно решать задачу регрессии с помощью рассматриваемой вероятностной модели. Обычно используются стационарные Гауссовские процессы, т.е. математическое ожидание является костантой, а ковариационная функция зависит лишь от расстояния между наблюдаемыми элементами выборки.

Пусть решается задача регресии для некоторой функции $f(\mathbf{x})$, заданной на вещественном пространстве $\mathbb{R}^d$, $f : \mathbb{R}^d \to \mathbb{R}$. Пусть также задана конечная

выборка $\mathcal{X} = \{(\mathbf{x}_k, y_k)\}_{i=1}^n$, где $\mathbf{x}_k \in \mathbb{R}^d$ – наблюдаемые элементы выборки, $\mathbb{R} \ni y_k = f(\mathbf{x}_k) + \varepsilon_k$, $\varepsilon_k \sim \mathcal{N}(0, \sigma_n^2)$ – зашумлённое значение целевой переменной (рассматривается независимый одинаково распределённый шум $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$). Далее обозначим $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times d}$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$. Тогда удобно моделировать f с помощью Гауссовского процесса. Предположим, что f – некоторый скрытый Гауссовский процесс (как пример рассмотрим несмещенный стационарный Гауссовский процесс):

$$f \sim GP(0, k(\cdot, \cdot)).$$

Также введём в рассмотрение матрицу ковариации на объектах выборки:

$$K(T, P) = \{k(t_i, p_j) | i = 1, \ldots, v;\ j = 1, \ldots, w\},\ T \in \mathbb{R}^{v \times d},\ P \in \mathbb{R}^{w \times d}$$

Вводятся скрытые переменные модели $\boldsymbol{f} = (f_1, f_2, \ldots, f_n) \in \mathbb{R}^n$ – истиные значения целевой переменной в рассматриваемых точках выборки. Тогда, если известно распределение на шумовую компоненту:

$$p(y_i | f_i) = \mathcal{N}(y_i | f_i, \sigma_i^2).$$

Совместное распределение на зашумленное и истинное значения будет иметь вид:

$$p(\boldsymbol{y}, \boldsymbol{f} | X) = p(\boldsymbol{f} | X) p(\boldsymbol{y} | \boldsymbol{f})$$

Теперь пусть дана тестовая выборка $X_* \in \mathbb{R}^{l \times d}$ и требуется оценить значение целевой переменной f_* на этих элементах. Тогда:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X, X_*) & K(X_*, X_*) \end{bmatrix}\right)$$

Учитывая шумовую компоненту на откликах $\boldsymbol{y}$:

$$\operatorname{cov}(\boldsymbol{y}) = K(X, X) + \sigma_n^2 I$$

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X, X_*) & K(X_*, X_*) \end{bmatrix}\right)$$

Тогда прогнозируемое условное распределенее будет иметь вид:

$$\boldsymbol{f_*} | \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{m_*}, \boldsymbol{K_*}).$$

Выражение для параметров распределения:

$$\boldsymbol{m_*} = \mathbb{E}(f_* | y) = K(X, X_*)^T (K(X, X) + \sigma_n^2 I)^{-1} \boldsymbol{y}$$

$$\boldsymbol{K_*} = \operatorname{cov}(f_* | y) = K(X_*, X_*) - K(X, X_*)^T (K(X, X) + \sigma_n^2 I)^{-1} K(X, X_*)$$

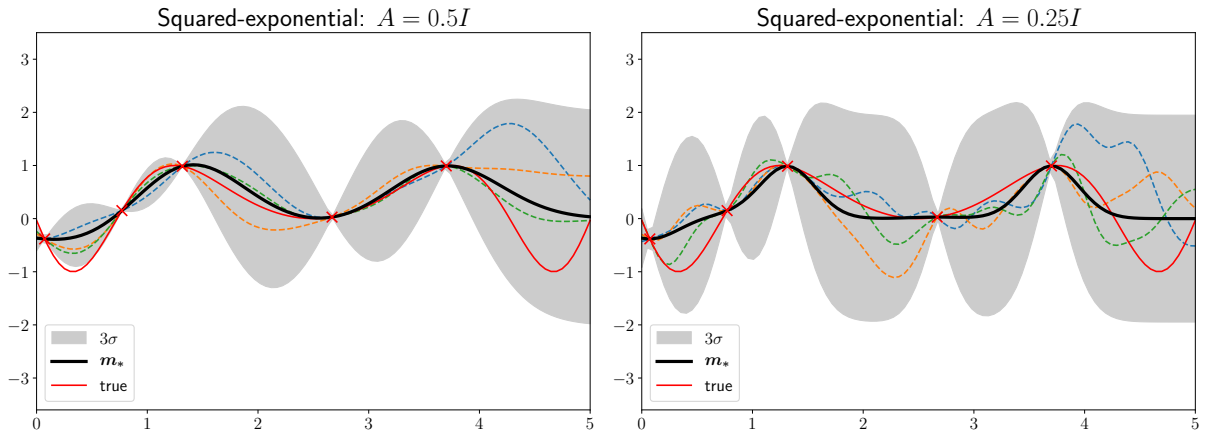


Рис. 1: Вид типичных реализаций Гауссовского процесса в зависимости от параметра A функции расстояния для ядра Squared-Exponential

Выбор ядра Вид ядра определяет структуру отклика, которые может выдавать модель. Например, если бы мы предполагали, что функция имеет периодический вид, то можно было бы рассматривать периодические ядра. Но как упоминалось ранее, обычно рассматриваются стационарные Гауссовские процессы. В таком случае вид ядра показывает насколько мы будем принимать близлежащие точки выборки в расчёт при предсказаниях.

Поскольку ковариационная функция зависит лишь от расстояния между объектами, будем писать $k(r(\cdot,\cdot))$, где r – некоторая функция расстояния между объектами (например, $r^2(\mathbf{x},\mathbf{x'}) = (\mathbf{x}-\mathbf{x'})^T A(\mathbf{x}-\mathbf{x'})$, где A – диагональная матрица)

Например, если рассмотреть ядро SE, имеющего колоколообразную форму, то в зависимости от вида функции расстояния будут менять вид типичные реализации Гауссовского процесса.

Квадратично-экспоненциальное ядро (SE):

$$k_{SE}(r) = \exp\left(-r^2\right),$$

Можно также рассматривать более общее семейство ядер Matérn:

$$k_{\text{Matérn}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}r\right)^{\nu} K_\nu\left(\sqrt{2\nu}r\right),$$

где K_ν – модифицированная функция Бесселя. Здесь параметр ν является настраиваемым.

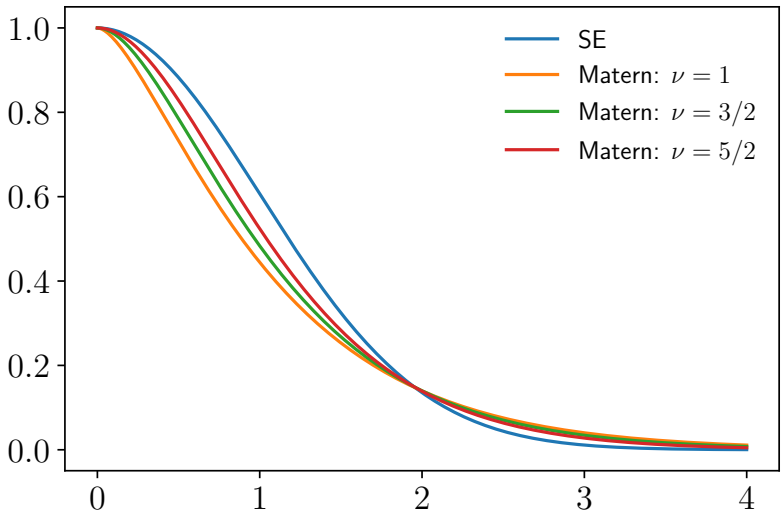


Рис. 2: Различие профилей функций из семейства Matérn

На практике обычно фиксируется параметр $\nu = \frac{5}{2}$ или $\nu = \frac{3}{2}$:

$$k_{\nu=3/2}(r) = \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right)$$

$$k_{\nu=5/2}(r) = \left(1 + \sqrt{5}r + 5r^2/3\right) \exp\left(-\sqrt{5}r\right)$$

Также существуют ядра для категориальных гиперпараметров. При этом в качестве функции расстояния можно использовать взвешенное расстояние Хэмминга[5]:

$$k_{cat}(\mathbf{x}_p, \mathbf{x}_q) = \exp\left[\sum_{i=1}^{d} -\lambda_i (1 - \delta(\mathbf{x}_p^i, \mathbf{x}_q^i))\right],$$

где параметры λ_i – настраиваемые, $\delta(\cdot, \cdot)$ – символ Кронекера.

Помимо этого можно использовать свойства ядер и строить их композиции:

$$k_{mix}(\mathbf{x}_p, \mathbf{x}_q) = k_{\text{Matérn}}(\mathbf{x}_p^{cont}, \mathbf{x}_q^{cont}) + k_{cat}(\mathbf{x}_p^{cat}, \mathbf{x}_q^{cat}),$$

где $\mathbf{x}^{cat}$ – категориальные составляющие вектора гиперпараметров, $\mathbf{x}^{cont}$ – непрерывные. Параметры ядра θ можно настраивать автоматически с помощью максимизации правдоподобия:

$$p(\boldsymbol{y}|X, \theta) = \int p(\boldsymbol{y}, \boldsymbol{f}|X, \theta) p(\boldsymbol{f}|X, \theta) d\boldsymbol{f} \longrightarrow \max_{\theta}$$

Далее обозначим матрицу ковариации, полученное через параметризуемую ковариационную функцию как $K_\theta(X, X)$. Поскольку нормально распределение принадлежит

экспоненциальному классу распределений, то интеграл выше может быть выражен аналитически. Тогда, логарифмируя нормальное распределение, имеем равносильную формулировку:

$$\log p(\boldsymbol{y}|X,\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\det\left(K_\theta(X,X) + \sigma_n^2 I\right)\right) - \frac{1}{2}\boldsymbol{y}^T(K_\theta(X,X) + \sigma_n^2 I)\boldsymbol{y}$$

$$\log p(\boldsymbol{y}|X,\theta) \longrightarrow \max_\theta$$

Также можно рассмотреть смещенную версию Гауссовского процесса с параметризованной функцией среднего m_θ, задающую отступ и аналогично настраивать её параметры.

3.3.2 Вспомогательные функции

Пусть выполняется n-ая итерация метода, тогда до этого были рассмотрены точки из $\mathcal{D}_n$ и требуется предоставить следующую точку для эксперимента $\mathbf{x}_{n+1}$. Рассмотрим функцию полезности $U : \mathbb{R}^d \times \mathbb{R} \times \Theta \to \mathbb{R}$. Она показывает качество эксперимента, то есть если бы была рассмотрена тройка $(\mathbf{x}, v, \theta)$, где $\mathbf{x}$ – рассматриваемая точка, $v = f(x)$ – соответствующее значение оптимизируемой функции в точке $\mathbf{x}$, θ – параметры используемой модели. Тогда, маргинализуя по параметрам θ и v можно получить ожидаемую полезность точки $\mathbf{x}$:

$$\alpha(\mathbf{x}; \mathcal{D}_n) = \mathop{\mathbb{E}}_{\theta} \mathop{\mathbb{E}}_{v|\mathbf{x},\theta} \left[U(\mathbf{x}, v, \theta)\right].$$

Далее именно эту функцию будем называть вспомогательной. Обычно при их оптимизации считают, что параметры модели θ фиксированы.

Существует много различных подходов к выбору функции полезности, рассмотрим некоторые из них.

Подходы, основанные на улучшении Предположим, что на текущей итерации лучшим значением является τ и решается задача максимизации. Тогда в качестве вспомогательной функции можно использовать вероятность улучшения (Probability of improvement). Заметим, что если бы Гауссовский процесс был выбран в качестве вероятностной модели, то вероятность улучшения записывалась как:

$$\alpha_{PI}(\mathbf{x}; \mathcal{D}_n) = \mathbb{P}\left[v > \tau\right] = \Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right),$$

заметим, что в данном случае $U(\mathbf{x}, v, \theta) = \mathbb{I}[v > \tau]$, а выбор следующей точки будет соответствовать моде рассматриваемого апостериорного распределения.

Также можно учитывать насколько сильно улучшилось бы значение рассматриваемой функции, в таком случае функцию полезности и соответствующую вспомогательную функцию можно записать в виде:

$$U(\mathbf{x}, v, \theta) = (v - \tau)\mathbb{I}[v > \tau]$$

$$\alpha_{EI}(\mathbf{x}; \mathcal{D}_n) = (\mu_n(\mathbf{x}) - \tau)\Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right) + \sigma_n(\mathbf{x})\phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right),$$

где $\phi(\cdot)$, $\Phi(\cdot)$ – соответственно функция плотности и функция распределения стандартного нормального распределения $\mathcal{N}(0, 1)$. Данная вспомогательная функция показывает ожидаемое улучшение (Expected-Improvement).

Подходы, основанные на оптимистическом предсказании В данном случае предполагается, что прогноз модели будет соответствовать оптимистичному варианту. Примером вспомогательной функции для такого подхода является верхняя граница гауссовского процесса (Gaussian process upper confidence bound):

$$\alpha_{UCB}(\mathbf{x}; \mathcal{D}_n) = \mu_n(\mathbf{x}) + \sqrt{\beta_n}\sigma_n(\mathbf{x}),$$

здесь параметр β_n – настраиваемый.

Подходы, использующие информационную ценность В данном случае учитываются апостериорные распределения по неизвестному минимизатору $\mathbf{x}^*$. Далее будем его обозначать как $p_*(\mathbf{x}|\mathcal{D}_n)$. Такие распределения неявно индуцируются апостериорными распределением на целевую переменную f. Рассмотрим используемого здесь метода – entropy serach (ES), главной задачей которого является уменьшение неопределённости в положении $\mathbf{x}^*$. В терминах функции полезности будем записывать это как:

$$U(\mathbf{x}, y, \theta) = H(\mathbf{x}^*|\mathcal{D}_n) - H(\mathbf{x}^*|\mathcal{D}_n \cap \{(x, y)\}),$$

где $H(\cdot)$ – это дифференциальная энтропия апостериорного распределения $p_*(\mathbf{x}|\mathcal{D}_n)$. Тогда вспомогательная функция будет иметь вид:

$$\alpha_{ES}(\mathbf{x}; \mathcal{D}_n) = H(\mathbf{x}^*|\mathcal{D}_n) - \mathop{\mathbb{E}}_{y|\mathcal{D}_n, \mathbf{x}} H(\mathbf{x}^*|\mathcal{D}_n \cup \{(x, y)\}),\ y \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x}) + \sigma^2).$$

Но поскольку данная функция непригодна для непрерывных областей поиска используется другой вариант записи для данной вспомогательной функции (Predictive Entropy Search):

$$\alpha_{PES}(\mathbf{x}; \mathcal{D}_n) = H(y|\mathcal{D}_n, \mathbf{x}) - \mathop{\mathbb{E}}_{\mathbf{x}^*|\mathcal{D}_n} H(y|\mathcal{D}_n, \mathbf{x}^*, \mathbf{x}^*),$$

здесь было использовано свойство симметричности для взаимной информации. В работе [3] указаны эффективные методы вычисления данного выражения.

3.3.3 Tree Parazen Estimators

Рассмотрим другую вероятностную модель [1], которая в отличие от Гауссовского процесса, где апостериорное распределение на функцию отклика $p(f|\mathbf{x})$ формируется напрямую, моделирует это распределение через $p(\mathbf{x}|f)$ и $p(f)$. В данном случае структурное пространство гиперпараметров описывается с помощью дерева. В каждом узле оно подменяет $p(x|y)$ непараметрическими распределениями, вид которых определяется видом гиперпараметра. Например, равномерное распределение заменяется усечённой смесью гауссиан, категориальное – перевзвешенным категориальным. Таким образом имеем:

$$p(\mathbf{x}|f, \mathcal{D}) = \begin{cases} l(\mathbf{x}) & \text{если } f < f^*, \\ g(\mathbf{x}) & \text{если } f \geqslant f^*, \end{cases}$$

где f^* – лучшее наблюдаемое значение минимизируемой функции f, а $l(\cdot)$, $g(\cdot)$ – заменяющие распределения. Далее фиксируется некоторая квантиль γ, такая что $P(f < f^*) = \gamma$ – условно «мягкость» модели, чтобы модель могла сэмплировать точки из $g(\mathbf{x})$. На априорное распределения $p(y)$ ограничения не накладываются.

Тогда можно записать выражение для:

$$\alpha_{EI} = \int_{-\infty}^{f^*} (f^* - f) p(f|x)\, df = \int_{-\infty}^{f^*} (f^* - f) \frac{p(\mathbf{x}|f)p(f)}{p(\mathbf{x})}\, df = \frac{\gamma f^* l(\mathbf{x}) - l(\mathbf{x}) \int_{-\infty}^{f^*} [f p(f)]\, df}{\gamma l(\mathbf{x}) + (1-\gamma) g(\mathbf{x})}.$$

Заметим, что:

$$\alpha_{EI} \propto \left(\gamma + \frac{g(\mathbf{x})}{l(\mathbf{x})}(1-\gamma)\right)^{-1},$$

откуда следует, что для того чтобы максимизировать α_{EI}, алгоритму будет выгоднее брать точки для которых отношение $g(\mathbf{x})/l(\mathbf{x})$ минимально.

4 Эксперименты

Предлагается сравнить представленные выше методы на качество и скорость работы. В данном случае под скоростью понимается количество итераций нужное для сходимости до примерного глобального минимума в сравнении с другими методами. Сравнение времени работы реализаций методов, используемых автором, с технической точки зрения было бы некорректным. Методы GP-EI, GP-PI, GP-UCB, TPE-EI исполняются, не используя промежуточных фаз, прямо из Python. Spearmint рассматривает каждую итерацию GP-PES как отдельную задачу, причём для каждой создаётся отдельный процесс. После этого данные заносятся в базу данных MongoDB. С учётом этого сравнение времени будет подразумевать сравнение верхних временных границ, которые могут сильно отличаются от точных значений.

Предлагается рассмотреть работу данных алгоритмов на функциях из стандартного набора для тестирования оптимизационных методов [12] и при подборе оптимальных параметров SVM, обучаемого на датасете MNIST [7]. В таблице 2 и на рисунке 2 представлены рассматриваемые функции.

Метод	Реализация
GP-PI, GP-EI, GP-UCB	Scikit-Optimize [9]
GP-PES	Spearmint [11]
TPE-EI	HyperOpt [6]

Таблица 1: Реализации методов.

Название и рассматриваемя область	Функциональное представление	Минимальное значение и точки оптимума
Branin Function $\mathbf{x} \in [-5, 10] \times [0, 15]$	$f(\mathbf{x}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + +10(1 - \frac{1}{8\pi}\cos(x_1))$	$f(\mathbf{x}_*) = 0.397887$ $\mathbf{x}_* \in \{(9.42478, 2.475), (\pi, -12.275), (-\pi, 12.275)\}$
Six-Hump Camel Function $\mathbf{x} \in [-3, 3] \times [-2, 2]$	$f(\mathbf{x}) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + +x_1x_2 + (-4 + 4x_2^2)x_2^2$	$f(\mathbf{x}_*) = -1.0316$ $\mathbf{x}_* \in \{(0.0898, -0.7126), (0.0898, -0.7126)\}$
Rosenbrock Function $\mathbf{x} \in [-5, 10]^d,\ d = 2$	$f(\mathbf{x}) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + +(x_i - 1)^2$	$f(\mathbf{x}_*) = 0$ $\mathbf{x}_* = (1, 1, \ldots, 1)$
Colville Function $\mathbf{x} \in [-10, 10]^4$	$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + +(x_1 - 1)^2 + (x_3 - 1)^2 + +90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + +(x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	$f(\mathbf{x}_*) = 0$ $\mathbf{x}_* = (1, 1, \ldots, 1)$
Easom function $\mathbf{x} \in [-100, 100]^2$	$f(\mathbf{x}) = -\cos(x_1)\cos(x_2) \times \times \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$f(\mathbf{x}_*) = -1$ $\mathbf{x}_* = (\pi, \pi)$
Griewank Function $\mathbf{x} \in [-2, 2]^d,\ d = 6$	$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$f(\mathbf{x}) = 0$ $\mathbf{x}_* = (0, 0, \ldots, 0)$

Таблица 2: Тестовые функции.

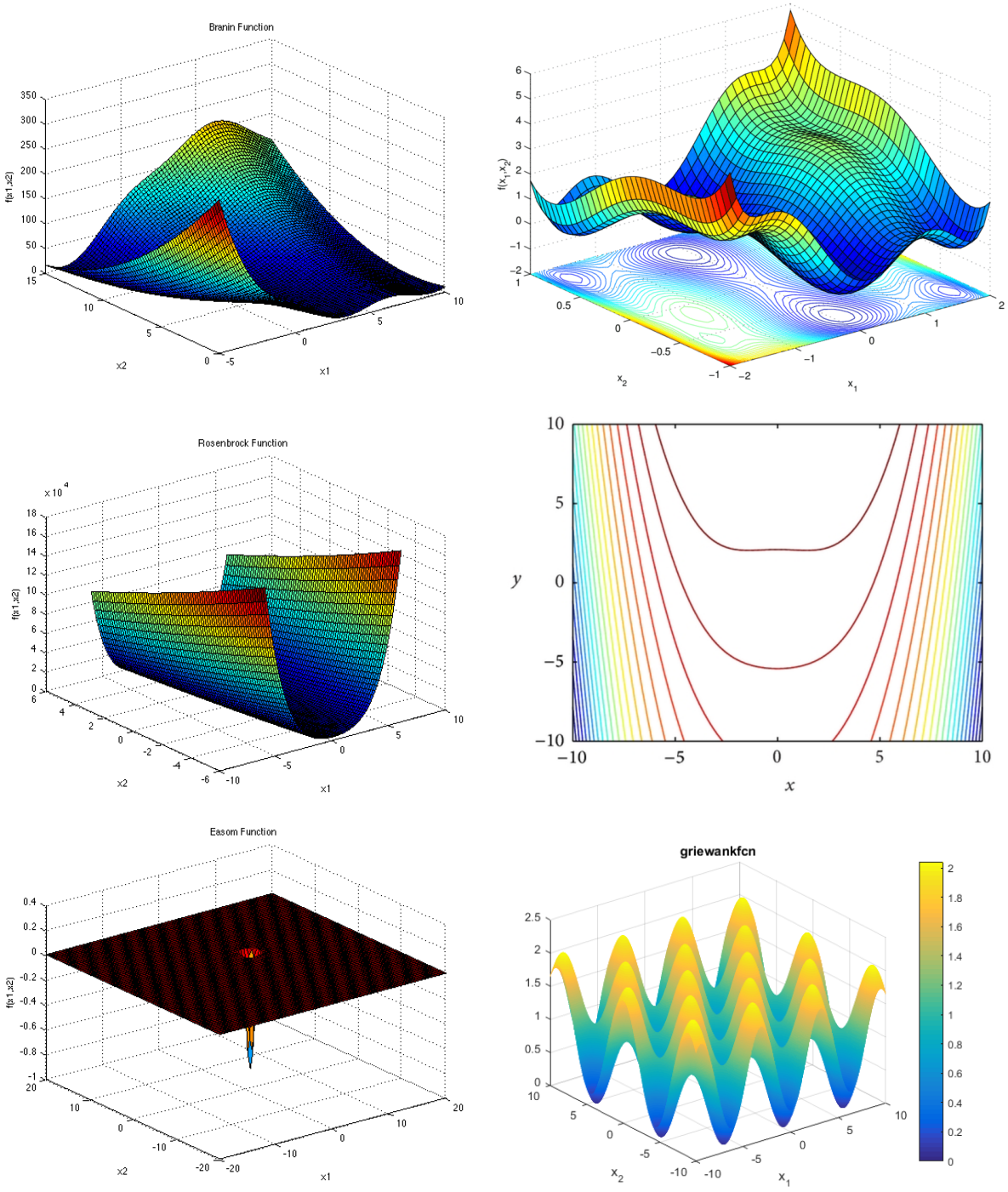


Рис. 3: Графики тестовых функций. Слева направо, сверху вниз это Rosenbrock Function, Six Hump Camel Function, Rosenbrock Function, Colville Function (линии уровня при $x_1 = x_3$, $x_2 = x_4$), Easom Fucntion, Griewank Function (при $d = 2$).

4.1 Эксперимент 1

Предлагается рассмотреть следующий сценарий: провести сравнение методов, основанных на гауссовском процессе в зависимости от вспомогательной функции. Рас-

сматривались функции: PI, ES, UCB ($\sqrt{\beta_n} = 1.96 = const$ – значение предлагаемое пакетом Scikit-Optimize по-умолчанию), PES. В данном эксперименте проводились вычисления на некоторых функциях из тестового набора, а также на их зашумлённых версиях. Использовался стандартный Гауссовский шум с параметрами $\mathcal{N}(0, 0.1)$ (моделям сообщалось соответствующее распределение). В качестве ядра Гауссовского процесса рассматривалось настраиваемое с помощью максимизации правдоподобия ядро Matérn 5/2. Вычислительные результаты представлены для 40 итераций соответствующего метода представлены на рисунках 4, 5, 6, 7.

В данном случае хуже всего с точки зрения и скорости и качества показала себя UCB. Если функция имеет слишком широкую область значений (как в стучае с Rosenbrock) метод даже не может приблизится к значению оптимума, а остаётся постоянным (параметр $\sqrt{\beta_n}$ – выбран слишком маленьким). Этого недостатка лишена функция PES, которая как раз-таки хорошо работает на таких функциях, даже при наличии шума (правда, в данном случае он существенен только при приближении к глобальному минимуму функции).

В представленных примерах функции EI и PI работали примерно схоже. Если их сравнивать, то можно видеть, что функция PI работает менее агрессивно (траектория почти гладкая).

Если сравнивать левые и правые графики, то они примерно одинаковы, хотя, это немного странно, если смотреть на вид функций (слишком резко меняются).

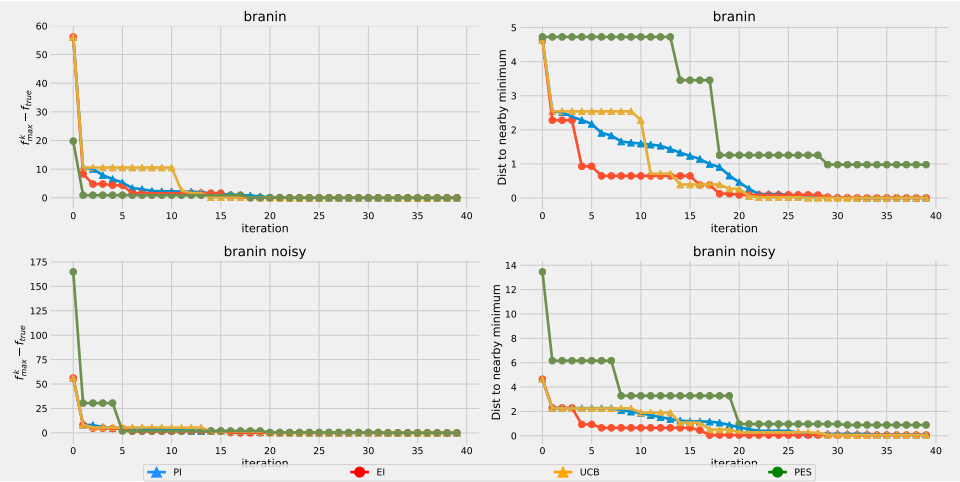


Рис. 4: **Эксперимент 1:** Branin Function

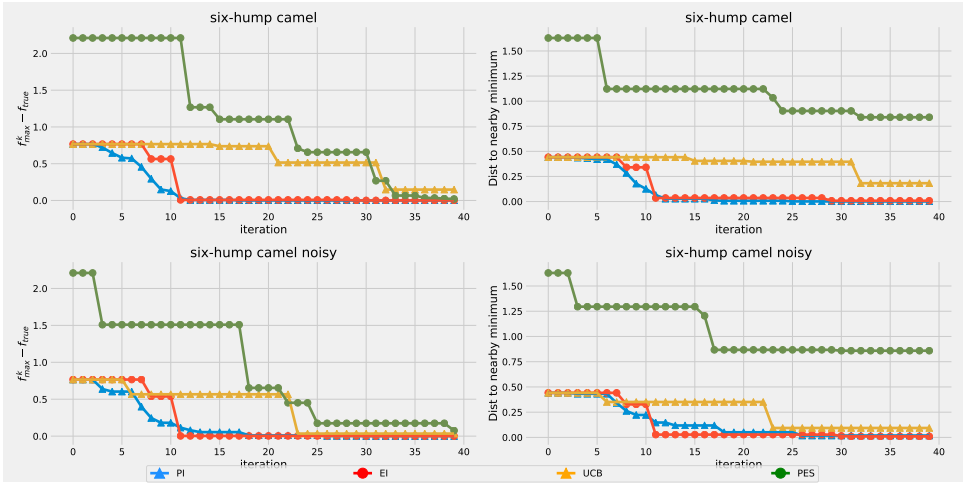


Рис. 5: **Эксперимент 1:** Six-Hump Camel Function

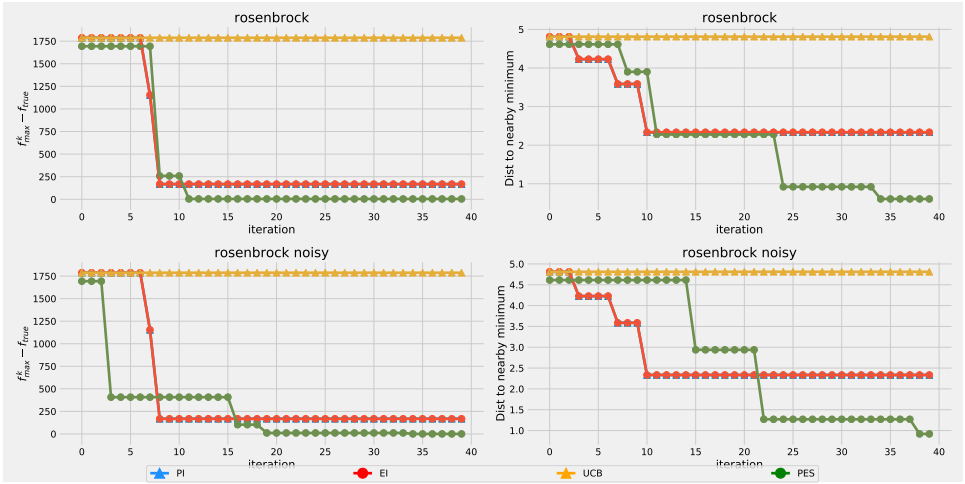


Рис. 6: **Эксперимент 1:** Rosenbrock Function

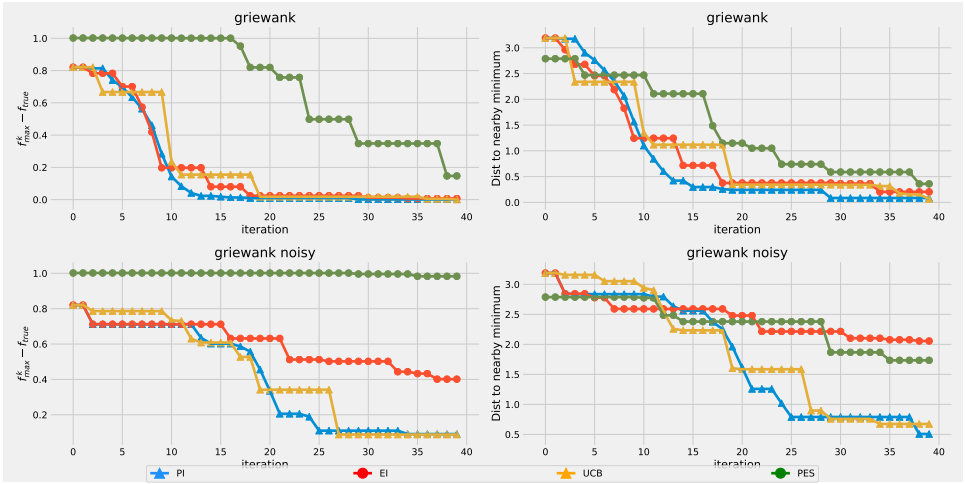


Рис. 7: **Эксперимент 1:** Griewank Function

4.2 Эксперимент 2

Предлагалось сравнить подходы в зависимости от структуры, строящегося вероятностного пространства. В качестве вспомогательной функции была взята Expected Improvement. В данном случае рассматривались следующие комбинации TPE-EI и GP-EI (с настраиваемым ядром Matérn 5/2). Результаты представлены на рисунках 8, 11, 9, 10. В данном случае нельзя выделить лучший метод. TPE-EI чаще меняет свои значения и чаще ближе к значению минимума в сравнении с GP-EI. Однако, метод GP-EI более устойчив к наличию шума в данных.

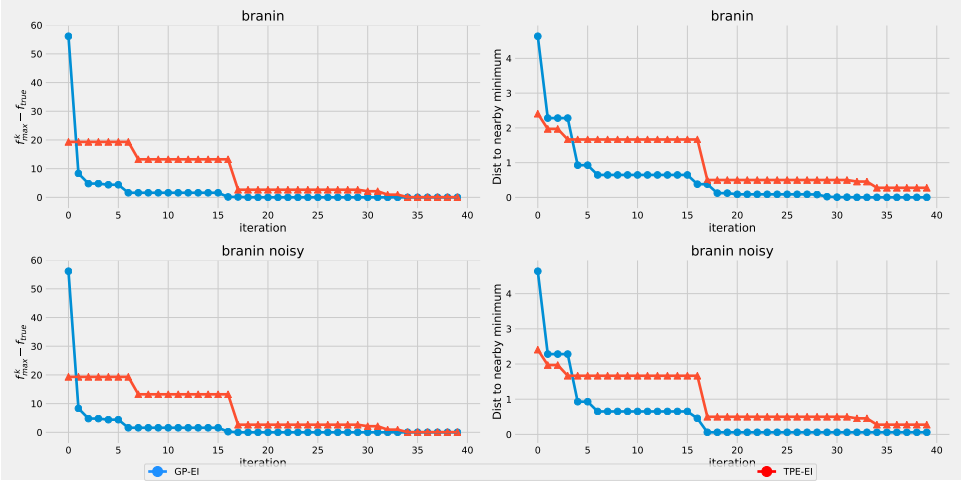


Рис. 8: **Эксперимент 2:** Branin Function

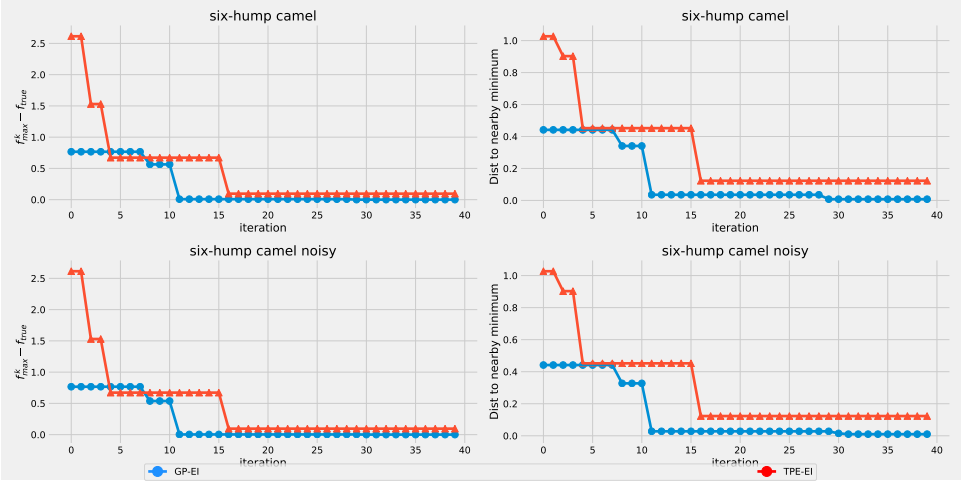


Рис. 9: **Эксперимент 2:** Six-Hump Camel Function

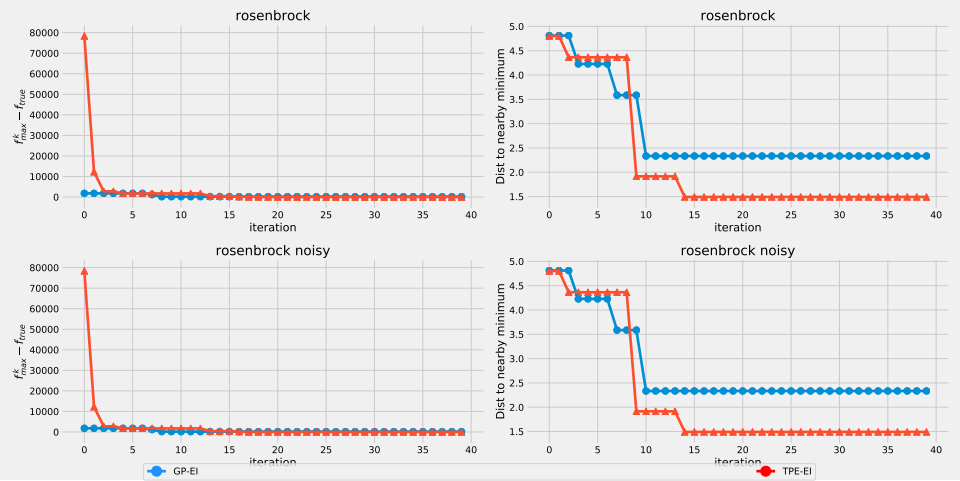


Рис. 10: **Эксперимент 2:** Rosenbrock Function

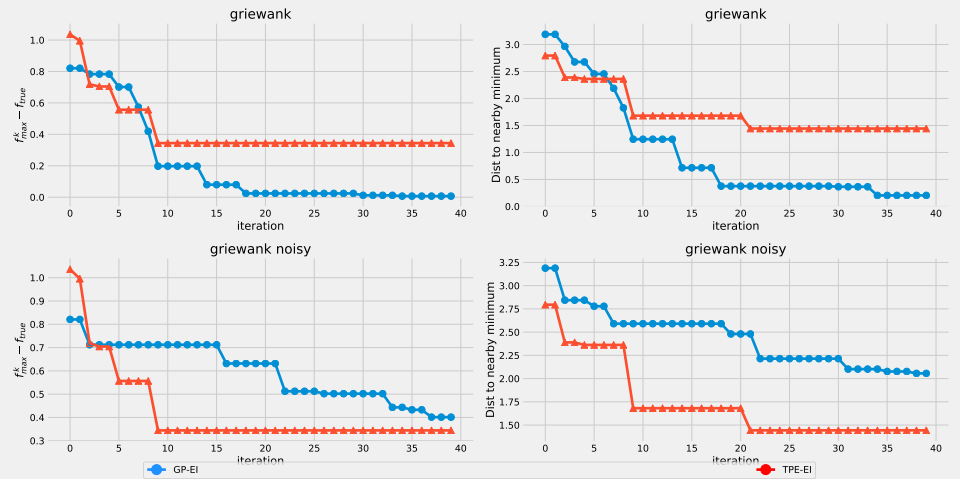


Рис. 11: **Эксперимент 2:** Griewank Function

4.3 Эксперимент 3

Предлагается исследовать влияние выбора ядра на качество и скорость сходимости. Рассматривался структура GP-EI. В качестве рассматриваемых ядер выбраны: настраиваемое и константные ядра Matérn 5/2 (MATERN5_UPD, MATERN5_FIX), Matérn 3/2 (MATERN3_UPD, MATERN3_FIX), Squared-Exponential (RBF_UPD, RBF_FIX). Результаты представленны на рисунках 12, 15, 13, 14, 16, 17.

Почти всегда методы с настраиваемыми параметрами работают лучше. Исключением является функция Easom, у которой один глобальный минимум, который максимизацей правдоподобия находится не будет, если не повезёт с выбором начальной точки, поскольку в области за кругом радиуса π в точке (π, π) она имеет значения

близкие к 0. Поэтому модель будет настраиваться почти на константное значение около нуля.

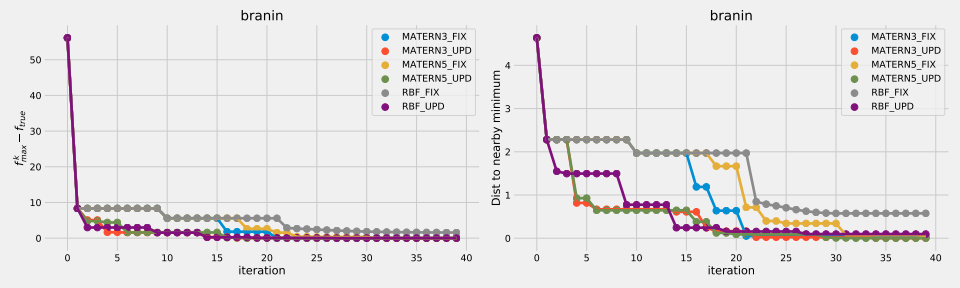


Рис. 12: **Эксперимент 3:** Branin Function

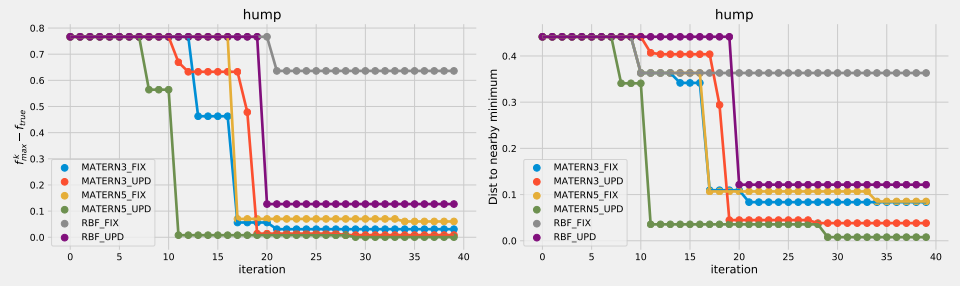


Рис. 13: **Эксперимент 3:** Six-Hump Camel Function

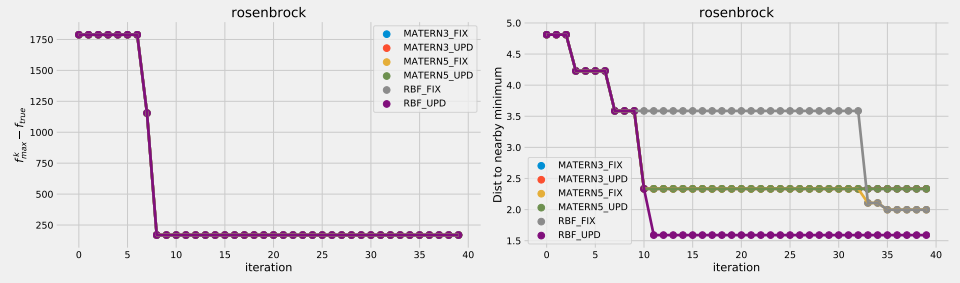


Рис. 14: **Эксперимент 3:** Rosenbrock Function

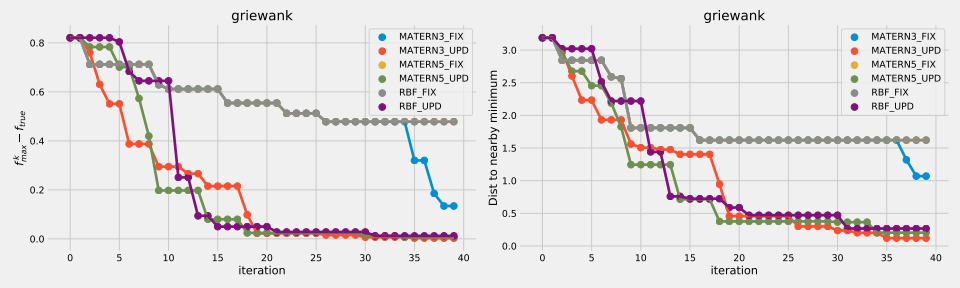


Рис. 15: **Эксперимент 3:** Griewank Function

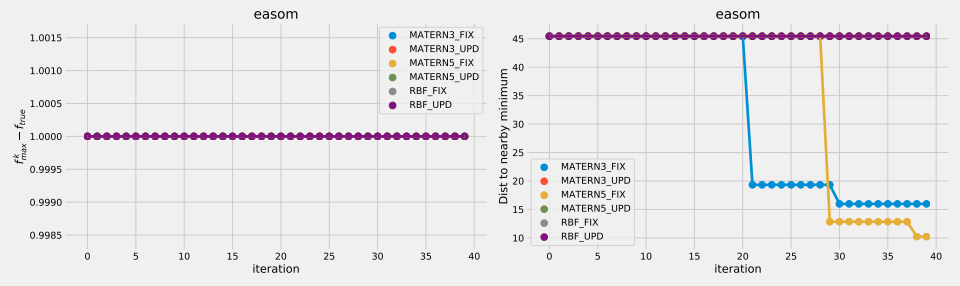


Рис. 16: **Эксперимент 3:** Easom Function

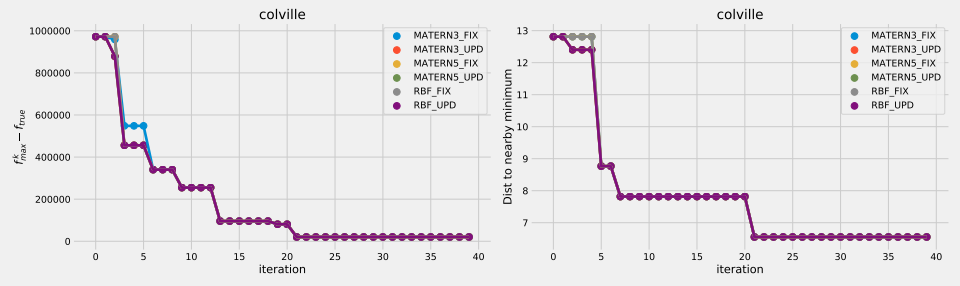


Рис. 17: **Эксперимент 3:** Colville Function

4.4 Эксперимент 4

Предлагается рассмотреть реальную задачу оптимизации гиперпараметров. В качестве обучаемой модели выбрана SVM. Рассматривается задача классификации изображений рукописных цифр из датасета MNIST. Каждое представляет из себя бинарную маску размера 28×28, на которых изображены 10 классов (цифры 0-9). Предлагается настраивать следующие параметры:

- Параметр C, отвечающий за штрафование ошибочных классификаций. Рассматривался отрезок $[1, 10^5]$;
- Параметр γ радиально базисного ядра. Рассматривался отрезок $[10^{-5}, 10^{-1}]$.

Предлагается настраивать точность (accuracy) модели на тестовой выборке. В качестве методов оптимизации гиперпараметров рассматривались GP-PI, GP-EI,GP-UCB, GP-PES, TPE-EI.

Как и утверждалось ранее метод GP-UCB показывает сравнимо худшие результаты на старте, однако в конце за счёт маленького коэффициента β_n настраивается лучше большинства методов (аналогия с learning-rate). GP-PES работает хорошо на старте, поскольку значение параметра C меняется в достаточно широком диапазоне. Методу TPE-EI повезло с выбором начального приближения, однако найти более лучшего претендента не удалось. Метод GP-EI хотя и сходился долго, но показал лучший результат.

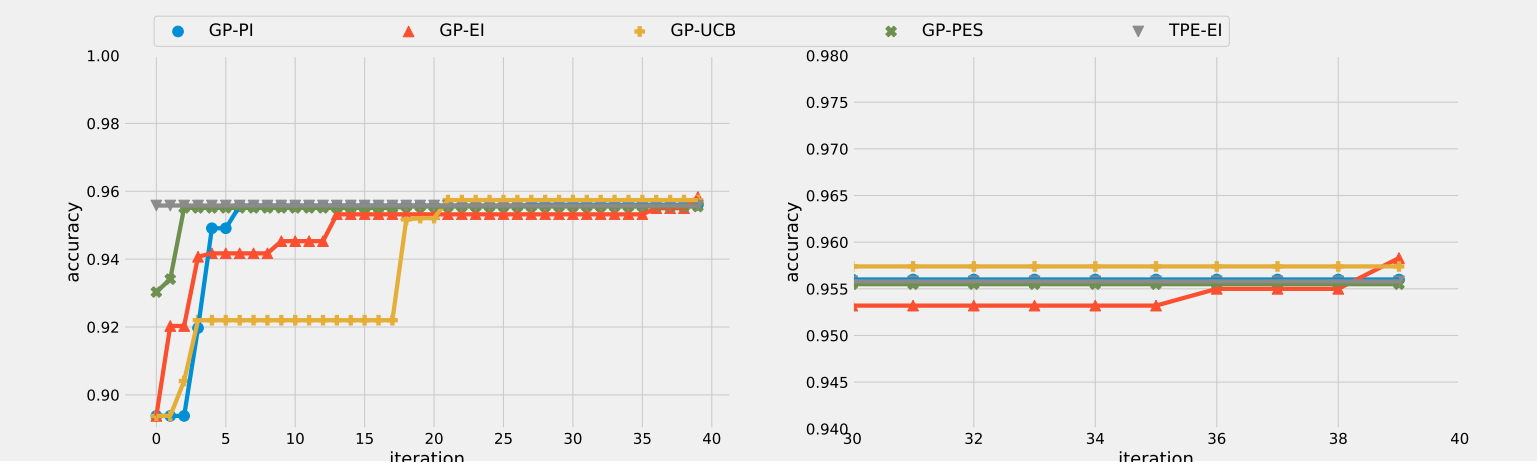


Рис. 18: **Эксперимент 4:** Классификация датасета MNIST с помощью SVM

4.5 Выводы из экспериментов

В результате экспериментов выявлено следующее:

- Выбор вспомогательной функции зависит от наличия шумовой компоненты в данных. При её наличии хорошим изначальным выбором будет Expected Improvement. В случае, если нужно донастроить модель лучше использовать UCB с маленьким шагом β_n;
- Выбор ядра должен основываться на виде данных, а сами ядра настраиваться в ходе обучения;
- Среди рассмотренных моделей выделить особой лучшей комбинации нельзя.

5 Заключение

В ходе работы были изучены основные подходы используемые при оптимизации гиперпараметров. Были соответствующие вычислительные эксперименты. В ходе экспериментов установлено, что среди рассмотренных методов нельзя выделить лучшую комбинацию, выбор параметров моделей должен основываться на данных.

В качестве дальнейшей работы в рамках Байесовской оптимизации можно рассмотреть методы, комбинирующие представленные вспомогательные функции (Portfolios of Acquisition Functions). Также можно рассмотреть усложнённые варианты вероятностных моделей, например случайный процесс Стьюдента [10].

Список используемой литературы

[1] Algorithms for hyper-parameter optimization / J. S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl // Advances in Neural Information Processing Systems 24 / Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett et al. — Curran Associates, Inc., 2011. — Pp. 2546–2554. http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

[2] *Florea A.-C., Andonie R.* Weighted random search for hyperparameter optimization. — 2020.

[3] *Hernández-Lobato J. M., Hoffman M. W., Ghahramani Z.* Predictive entropy search for efficient global optimization of black-box functions. — 2014.

[4] *Hutter F., Hoos H., Leyton-Brown K.* An efficient approach for assessing hyperparameter importance // Proceedings of International Conference on Machine Learning 2014 (ICML 2014). — 2014. — Junuary. — Pp. 754–762.

[5] *Hutter F., Hoos H. H., Leyton-Brown K.* Sequential model-based optimization for general algorithm configuration // Proceedings of the 5th International Conference on Learning and Intelligent Optimization. — LION'05. — Berlin, Heidelberg: Springer-Verlag, 2011. — P. 507–523.

[6] Hyperopt: Distributed asynchronous hyper-parameter optimization. http://hyperopt.github.io/hyperopt/.

[7] *LeCun Y., Cortes C.* MNIST handwritten digit database. — 2010. http://yann.lecun.com/exdb/mnist/.

[8] *Rasmussen C. E., Williams C. K. I.* Gaussian Processes for Machine Learning. — The MIT Press, 2006.

[9] scikit-optimize: Sequential model-based optimization in python. https://scikit-optimize.github.io/stable/.

[10] *Shah A., Wilson A., Ghahramani Z.* Student-t processes as alternatives to gaussian processes // Artificial intelligence and statistics. — 2014. — Pp. 877–885.

[11] Spearmint bayesian optimization codebase. https://github.com/HIPS/Spearmint/tree/PESM.

[12] *Surjanovic S., Bingham D.* Virtual library of simulation experiments: Test functions and datasets. — Retrieved June 6, 2020, from http://www.sfu.ca/~ssurjano.