Использование аппроксимаций матрицы гессиана функции ошибки при настройке параметров регрессионных моделей

Беляев Михаил[1,2,3] Бурнаев Евгений[1,2,3] Любин Александр[1,2]
mikhail.belyaev@datadvance.net, evgeny.burnaev@datadvance.net, alexander.lyubin@datadvance.net
1. Институт Проблем Передачи Информации,
127994, г.Москва, ГСП-4, Большой Каретный переулок, 19, стр.1
2. DATADVANCE,
105064, г.Москва, Садово-Черногрязская улица, 13/3
3. PREMOLAB,
141700, Московская область, г. Долгопрудный, Институтский переулок, 9

Аннотация

В работе рассматривается подход к ускорению вычисления матрицы гессиана, используемой при подстройке параметров аппроксиматора многомерных зависимостей на основе разложения по словарю базисных функций. Ускорение достигается за счет вычисления данной матрицы по неполной выборке данных. Основное внимание уделено методике выбора такой подвыборки.

1. Введение

Задача построения аппроксимации неизвестной зависимости, т.е. построения функции, приближенно описывающей поведение неизвестной зависимости, получила достаточно широкое распространение. В частности, она активно используется в метамоделировании при создании сложных инженерных объектов [1]. Аппроксимация строится по некоторому неполному набору данных: значения функции известны только на конечном множестве точек из пространства признаков (обучающей выборке).

Наиболее типичные методы, которые используются для решения этой задачи — это линейная регрессия, однослойная нейронная сеть, радиальные базисные функции [2]. В данной работе рассматривается аппроксиматор, построенный на основе разложения по словарю базисных функций различного типа [3]. В такой постановке задача сводится к нахождению оптимального набора параметров, характеризующих модель, с целью приближения истинной зависимости наилучшим образом. В качестве меры точности приближения используются различные функционалы ошибки, например сумма квадратов отклонений модели от истинных значений в точках обучающей выборки.

Подбор параметров такого аппроксиматора — сложная оптимизационная задача [4], предъявляющая к используемым алгоритмам оптимизации жесткии требования не только на точность нахождения минимума и скорость сходимости, но и на устойчивость к локальным экстремумам и временные затраты на одну итерацию. В рассматриваемом подходе предлагается использовать последовательную схему из алгоритмов первого и второго порядка [5]. Однако, использование алгоритмов второго порядка (таких как Trust Region) требует вычисление матрицы гессиана. Как будет показано ниже, в случаях больших объемов обучающих выборок именно вычисление этой матрицы может быть связано с основными временными затратами.

Таким образом, цель данной работы состоит в том, чтобы предложить эффективные методы аппроксимации матрицы гессиана, позволяющие сократить время вычисления и приводящие к не слишком большим потерям в точности итоговой аппроксимации.

2. Постановка задачи аппроксимации

Под задачей аппроксимации подразумевается приближенное восстановление функции по выборке данных. Пусть $y = f(x)$ — некоторая функция, которая определена и непрерывна на компакте $D \in R^n$. В общем случае функция f задает отображение из R^n в R^m, но без ограничения общности можно положить $m = 1$. Задача состоит в нахождении приближения этой функции по обучающей выборке данных

$$S_{learning} = \{x_i, y_i\}, \;\; x_i \in D, \;\; f(x_i) = y_i, i = 1...N_{learning}.$$

При этом мы требуем, чтобы аппроксимация бы-

ла близка к истинной функции $\hat{f} \approx f$, где за $\hat{f}$ обозначена найденная аппроксимация истинной функции f .

В качестве меры близости $\hat{f}$ и f можно использовать различные нормы, но самой удобной в практических целях и наиболее часто используемой является квадрат невязки:

$$Q(S, \hat{f}(\vec{x})) = \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \hat{f}(\vec{x}_i)\right)^2$$

где $N = \#S$ - мощность множества S. В дальнейшем будем называть $Q(S, \hat{f}(x))$ функцией ошибки на множестве S, или просто ошибкой.

3. Модель аппроксимации

В данной работе рассматривается разложение функции по словарю параметрических базисных функци, т.е.:

$$\hat{f}(\vec{x}) = \sum_{j=1}^{p} \alpha_j \psi_j(\vec{\theta}_j, \vec{x}) + \alpha_0 \qquad (1)$$

где $\{\alpha_j\}$, $j = 0, ..., p$ — весовые коэффициенты, $\{\psi_j\}$, $j = 1, ..., p$ — набор из p базисных функций, по которым осуществляется разложение, θ_j — параметры этих базисных функций. В общем случае, это могут быть функции разного вида. В данной работе мы рассматриваем три типа базисных функций:

- Сигмоидальные базисные функции

$$\psi_j\left(\vec{\theta}_j, \vec{x}\right) = \sigma\left(\vec{x}'\vec{\theta}_j + \theta_0\right) \qquad (2)$$

  Одномерная функция $\sigma(z)$ — это сигмоида, она должна монотонно возрастать, быть ограниченной сверху и снизу. Например, это может быть гиперболический тангенс: $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.

- Радиальные базисные функции

$$\psi_j\left(\vec{\theta}_j, \vec{x}\right) = exp\left(-\frac{\|\vec{x} - \vec{\theta}_j\|^2}{\theta_{j0}^2}\right). \qquad (3)$$

- Линейные функции

$$\psi_j\left(\vec{\theta}_j, \vec{x}\right) = X_{\theta_j}.$$

В дальнейшем мы будем придерживаться матричной записи формулы (1):

$$\hat{f}(\vec{x}) = \Psi(\Theta, \vec{x})\vec{\alpha}, \quad \vec{\alpha} = \{\alpha_j\}_{j=0}^{p}, \quad \Theta = \{\vec{\theta}_j\}_{j=1}^{p} \qquad (4)$$

Вектор-строка $\Psi(\Theta, \vec{x})$ состоит из значений функций словаря в точке $\vec{x}$ (а $\psi_0 = 1$ соответствует α_0). Таким образом, аппроксимирующая функция $\hat{f}(x)$ задается матрицей параметров базисных функций Θ и вектором коэффициентов разложения α. Далее аппроксимация будет обозначаться $\hat{f}(\theta, \alpha)$, а переменная $\vec{x}$ будет опускаться.

Аппроксимация, построенная подобным образом, позволяет приближать достаточно широкий класс функций f. В работах [6, 7] показано, что для произвольной $\hat{f}$, состоящей из p сигмоидальных функций, оценка точности аппроксимации имеет вид $O\left(\frac{1}{p^{\frac{1}{n}}}\right)$. Однако, если для формирования словаря использовать не произвольные функции, а выбирать их с учетом аппроксимируемой функции (настраивать их на f), то справедлива более сильная оценка $O\left(\frac{1}{p}\right)$.

Алгоритм построения аппроксиматора подробно описан в [5], но кратко может быть записан следующим образом:

1. Выбор размера словаря p.
2. Инициализация параметров аппроксиматора: θ и α.
3. Минимизация функции ошибки $Q(S_{learning}, \hat{f}(\vec{x}))$ путем подстройки параметров методом первого порядка.
4. Минимизация функции ошибки $Q(S_{learning}, \hat{f}(\vec{x}))$ путем подстройки параметров методом второго порядка.

В данной работе внимание уделено непосредственно пункту 4, хотя, безусловно, первые два пункта оказывают существенное влияние на возникающую задачу оптимизации. Более того, возможность применения предложенных методов, как будет показано позже, сама по себе зависит от выбранного числа базисных функций p.

4. Задача оптимизации параметров аппроксиматора

Прежде всего, стоит отметить возможность разделимости оптимизируемых переменных. При фиксированных значениях параметров θ, задача нахождения оптимальных весовых коэффициентов α аналогична стандартной задаче метода наименьших квадратов (МНК) или гребневой регрессии, если вводится штраф для борьбы с плохой обусловленностью. Тогда, схема оптимизации может выглядеть следующим образом: после каждой итерации алгоритма оптимизации параметров θ, происходит пересчет новых весовых коэффициентов α.

С другой стороны, оба типа параметров могут рассматриваться как один набор параметров нелинейной модели $\omega = \{\alpha; \theta\}$, тогда оптимизация идет

по всему набору параметров. Сравнение данных методов приведено в работе [4], также там можно найти конкретный вид градиентов и матриц вторых производных, используемых в задаче оптимизации.

Здесь же мы остановимся только на представлении всех параметров в виде одного вектора, т.к. данный вид проще для изложение, а полученные результаты применимы в обоих случаях. Итак, введем обозначения:

- $\vec{e}(\Theta,\alpha) = \hat{f}(X,\Theta,\alpha) - y$ — вектор невязок на выборке S_{train}.
- $\omega = \{\omega_1, \omega_2, ..., \omega_p\}$ - вектор, состоящий из всех элементов матрицы Θ и вектора $\vec{\alpha}$.

Тогда функция ошибки будет иметь вид:

$$Q(\vec{\omega}) = \frac{1}{2}\vec{e}(\vec{\omega})^T\vec{e}(\vec{\omega}). \quad (5)$$

Выпишем формулы для градиента и матрицы вторых производных (гессиана) функции ошибки по параметрам аппроксиматора. Вектор первых производных (символ вектора у ω и других величин будет опускаться, когда они будут использоваться в нижнем индексе):

$$Q_\omega = \vec{e}^T\vec{e} = \hat{f}_\omega^T\vec{e} = J^T\vec{e}.$$

где $\hat{f}$ — вектор значений аппроксимации в точках обучающей выборки; $J = \hat{f}_\omega$ — матрица якобиана, в каждой строке которой содержится вектор производных $\hat{f}(\vec{\omega},\vec{x})$ в точке $\vec{x}_i$ обучающей выборки.

Матрица вторых производных:

$$Q_{\omega\omega} = \vec{e}_\omega^T J + \vec{e}_\omega^T J_\omega = J^T J + \sum_{i=1}^{N} e_i \hat{f}_{\omega\omega}(x_i). \quad (6)$$

При дифференцировании во втором слагаемом возникают трехмерные массивы, для которых операция матричного умножения не определена.

Так как $\vec{e}$ — это вектор невязок, то можно положить их малыми и пренебречь вторым слагаемым в (6). Тогда приближенная формула для подсчета матрицы гессиана будет иметь вид

$$Q_{\omega\omega} \approx J^T J. \quad (7)$$

Задачи, которые могут быть представлены в виде (5), в теории оптимизации называются задачами о наименьших нелинейных квадратах. Наиболее распространенные подходы к решению задач такого рода — это методы Гаусса-Ньютона и Левенберга-Марквардта [8]. Первый из них отличается от стандартного метода Ньютона лишь использованием формулы (7) для подсчета матрицы вторых производных, т.е направление шага на очередной итерации вычисляется по формуле $\vec{d}^k = -(J^T J)^{-1} J^T e$. В методе Левенберга-Марквардта к матрице гессиана добавляется некоторая положительно определенная поправка Λ, что приводит к модифицированной формуле для подсчета шага:

$$\vec{d}^k = -(J^T J + \Lambda)^{-1} J^T e \quad (8)$$

4.1. Временные затраты одной итерации

Рассмотрим вычислительную сложнось одной итерации алгоритма оптимизации. Для этого сначала определим размеры всех матриц, входящих в выражение (8):

- матрица Якобиана $[J] = N * p(d_x + 1)$
- матрица $\left[J^T J\right] = p(d_x + 1) * p(d_x + 1)$

где p - число фуекций в линейном разложении, d_x - размерность пространства признаков, выражение [ ] обозначает размеры матрицы-аргумента.

Тогда, легко видеть, что стоимость вычисления матрицы $J'J$ равна $O\left(Np^2(d_x+1)^2\right)$, обращение матрицы $J^T J + \Lambda$ стоит $O\left(p^3(d_x+1)^3\right)$. Для сравнения приведем значения N, p и d_x , которые часто используются при построении аппроксимаций:

- $N = 10^5$,
- $p = 100 - 200$,
- $d_x = 9$.

Следовательно, даже на таком наборе параметров видно, что вычисление матрицы $J'J$ оказывается в $50 - 100$ раз тяжелее ее обращения. Выходом из данной ситуации может стать использование не всей обучающей выборки, а лишь ее части объемом βN, где $\beta \in [0;1]$. Однако, данный подход поднимает два важных вопроса:

1. Как производить выбор точек, по которым производится подсчет гессиана
2. Как определить степень прореживания.

К сожалению, авторам не удалось найти работы, в которых поднимались и решались аналогичные вопросы. Поэтому были разработаны несколько принципиально различных подходов, представленных ниже. Первый вопрос, который возник при решении данной проблемы, состоит в частоте обновления подвыборки, по которой идет оптимизация.

Экспериментальные данные подтвердили, что использование одной подвыборке на большом количестве итераций, приводит к двум эффектам:

- Даже если подвыборка изначально выбрана оптимальным образом (со строго сформулированным понятием оптимальности), после некоторого числа итераций, ее свойство оптимальности теряется.

- Ошибка на остальных точках обучающей выборки либо остается неизменной, либо увеличивается с номером итерации.

Данные особенности можно объяснить тем, что аппроксимация меняется довольно сильно с номером итерации и, соответственно, в силу вступают эффекты переобучения [2]. В связи с этим предлагается использовать регулярно обновляющийся набор точек из исходной обучающей выборки. В частности, обновлять набор на каждой итерации обучающего алгоритма.

4.1.1. Выбор случайной подвыборки. Идея данного метода состоит в выборе произвольной подвыборки на каждой итерации алгоритма оптимизации. Очевидны преимущества данного подхода:

- Быстрота выбора подмножества.
- Равномерность заполнения пространства D точками подвыборок при росте числа итераций.

С другой стороны, недостатки данного алгоритма тоже очевидны:

- При неудачном выборе подвыборки процесс оптимизации может существенно отклониться от оптимальной траектории.
- Выбор точек никак не обусловлен оптимизируемой функцией.

4.1.2. Выбор подвыборки с максимальным отклонением аппроксимации. Данный метод предполагает оценивать матрицу гессиана на каждой итерации по βN точкам с максимальными невязками (отклонением аппроксимации от истинного значения функции).

Пусть алгоритм оптимизации выполнил $i-1$ итерацию, текущая аппроксимация $\hat{f}_i(x) = \vec{\psi}(\theta_i, \vec{x})\alpha_i$, где индекс i относится к номеру итерации. Выберем из вектора $\vec{e} = \hat{f}(\vec{x}) - f(\vec{x})$ невязок βN максимальных по абсолютному значению элементов. Используем точки, соответсвующие этим элементам, для вычисления матрицы гессиана и оценке шага алгоритма оптимизации $\vec{d_i}$.

4.1.3. Приближение диагонали гессиана. Обозначим за H_N - матрицу гессиана, построенную по N точкам, те $H_N = Q_{\omega\omega}$ в обозначениях (7). Рассмотрим дополнительную точку x_{N+1} и подсчитаем гессиан H_{N+1} . Из аддитивного вида (5) и (7) легко заключить, что $H_{N+1} = H_N + h(x_{N+1})$, где за $h(x_{N+1})$ обозначена добавка, вносимая отдельной точкой. Таким образом

$$H_N = Q_{\omega\omega} = \sum_{i=1}^{N} h(x_i) \tag{9}$$

Теперь обратим внимание на то, что диагональ Гессина вычисляется за $O(Np(d+1))$ операций. Воспользуемся этими двумя фактами и попытаемся найти наилучшую комбинацию βN из N точек, приближающих диагональ. Задача сводится к следующей: пусть имеется вектор $\vec{a} = \sum_{i=1}^{N} \vec{a}_i$. Необходимо найти такой набор индексов $\Omega \subseteq \{1,\ldots,N\}$, $\#(\Omega) = \beta N$, что сумма $\sum_{i\in\Omega} \vec{a}_i$ наилучшим образом приближает исходный вектор $\vec{a}$.

К сожалению, данная задача не имеет аналитического решения. Более того, сама эта задача должна решаться за крайне ограниченное время. Поэтому было предложено использовать жадный алгоритм: выбирать те вектора, которые по норме L_2 ближе всех к исходному вектору $\vec{a}$.

4.2. Выбор оптимальной степени прореживания

Еще один вопрос, требующий решения, когда речь идет о подсчете гессиана по неполной выборке данных — это выбор степени прореживания, т.е. числа точек, по которым происходит подсчет этой матрицы. Для решения этой проблемы было предложено использовать в качестве нормы близости приближенной матрицы к ее истинному значению L_2 норму близости их диагоналей. Данный выбор обусловлен двумя факторами. Во-первых, как отмечалось выше, расчет диагонали матрицы гессиана — задача вычислительно легкая. И во-вторых, данный подход позволяет избежать дополнительных затрат на обращения матриц (в отличае от сравнения L_2 норм матриц). Кроме того, эксперименты показали, что с точностью до шума, обусловленного рандомизацией в выборе подвыборок, данная функция является монотонной при изменении коэффициента β от 0 до 1.

Выбор степени прореживания предлагается делать один раз перед началом процесса оптимизации. Для этого задаемся уровнем допустимого отклонения истинной матрицы от приближенного значения и ищем такое наименьшее β_{min} , которое дает заданную точность.

Стоит отметить, что в случае рандомизированных алгоритмов формирования подвыборки (см пункт 4.1.1), отклонение приближения от истинного значения объясняется не только степенью прореживания, но и случайностью в выборе подмножества. Поэтому, логично проводить усреднение по нескольким запускам при заданном уровне прореживания β.

5. Описание экспериментов

Тестирование производилось на различных выборках реальных и искуственных задач размером от 15 000 точек до 200 000 точек [11], [10]. Размерность

выборок варьровалась от 2 до 20. Обратим внимание, что размерность 20 при количестве регрессоров p = 200 является границей разумности использования прореживания гессиана. При больших значениях произведения $p(d_x+1)$ операция обращения матрицы гессиана становится сравнимой в плане вычислительных затрат с её вычисления.

На рис.1 приведен пример двумерной реализации функции rastrigin.

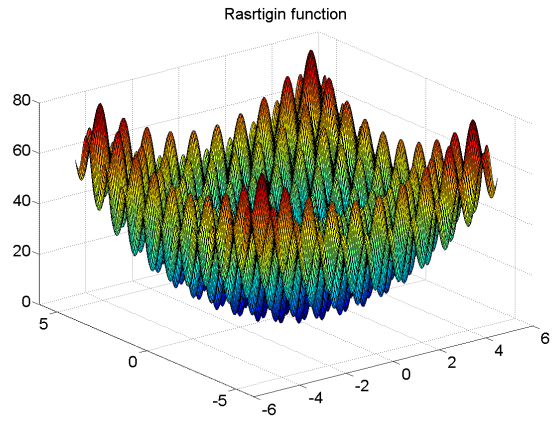

Рис. 1. Функция Rastrigin

Для чистоты эксперимента во всех метода предполагался один и тот же уровень прореживания $\beta = 0.3$.

5.1. Кривые Долана-Мора

Одним из наиболее удобных инструментов для сравнения результатов работы аппроксиматоров на большом числе тестовых выборок является графическое представление в виде кривых Долана Мора [9]. По оси абсцисс данных графиков откладывается отношение $p = \frac{Q_i}{Q_{min}}$ ошибки рассматриваемого (i-го) аппроксиматора к минимальной ошибке среди всех методов на конкретной задаче. По оси ординат откладывается доля задач, на которых ошибка метода, нормированная на минимальное значение ошибки среди всех методов, не превосходит величины p.

Таким образом, при $p = 1$ кривые отображают долю задач, на которых данный метод сработал лучше других, при $p = 10$ — доля задач, на которых метод сработал так, что ошибка аппроксимации не превосходит более чем в 10 раз наилучшую достижимую среди всех методов и т.д.

Отметим важные свойства данных кривых:

- Чем выше расположена кривая, тем лучше работает метод, соответствующий данной кривой.
- Точка по оси абсцисс, на которой кривая выходит на значение, равное 1, соответствует максимальному проигрышу метода всем другим методам на всех выборках.

5.2. Результаты численных экспериментов

Рис.2-4 отражают кривые Долана Мора для задач объема 15 000, 30 000 и 50 000 точек соответственно. Розовым цветом обозначены кривые, соответствующие обучению по полной выборке данных. Голубой цвет соответствует аппроксиматорам, на которых не использовались методы второго порядка (обучение останавливалось только на методе первого порядка). Синий цвет - приближение диагонали матрицы гессиана, зеленый - использование точек с максимальной ошибкой, красный - случайная подвыборка.

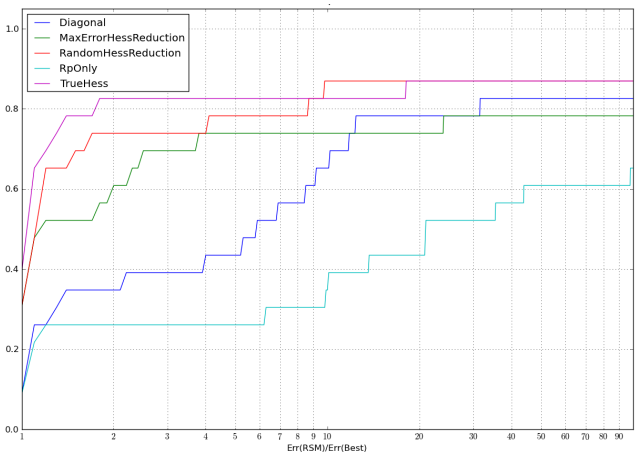

Рис. 2. Кривые Долана-Мора, выборки 15 000 точек

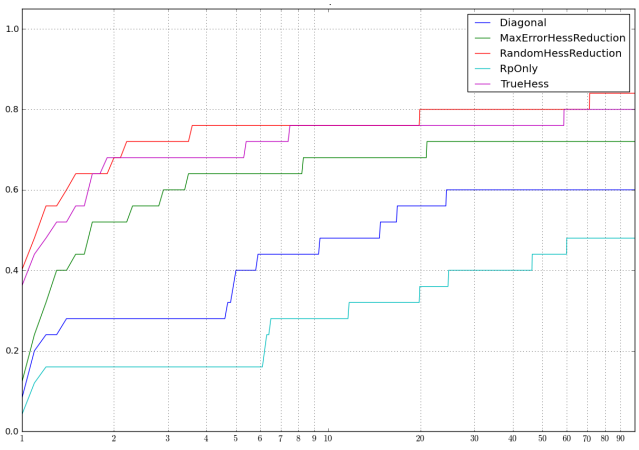

Рис. 3. Кривые Долана-Мора, выборки 30 000 точек

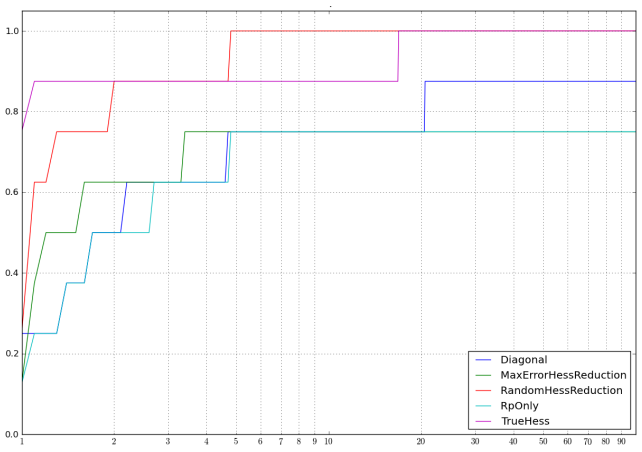

Рис. 4. Кривые Долана-Мора, выборки 50 000 точек

Полученные данные говорят о том, что наилучшим методом выбором подвыборки для подсчета матрицы гессиана, среди перечисленных выше, является метод выбора случайной подвыборки. Более того, на некоторых задачах эксперименты показали, что использование неполной выборки дает лучшие результаты, чем обычный метод. Данный эффект мы объясняем двумя фактами.

Во-первых, использование случайной подвыборки приводит к тому, что весь процесс оптимизации параметров движется по отличной от истинного процесса траектории, и достижение лучшей ошибки не противоречит никаким фактам и теоретическим результатам. Во-вторых, при достижении высокой точности аппроксимации, существенным фактором, влияющим на результаты экспериментов, становится машинный шум. В данном случае, даже выкидывание части информации (путем отбрасывание части выборки) приводит к меньшим эффектам, чем влияние шума.

6. Выводы

Данная работа показала принципиальную возможность ускорения процесса оптимизации параметров аппроксиматора на основе разложения по словарю параметрических функций. Ускорение достигается за счет уменьшения числа точек, по которым идет расчет матрицы гессиана. Показано, что уменьшение числа точек незначительно увеличивает ошибку на независимой тестовой выборке по сравнению с использованием истинной матрицы, посчитанной по всей доступной выборке данных.

Одним из самых эффективных методов является выбор подвыборок случайным образом. Дальнейшие работы могут быть связаны с попытками теоретического обоснования сходимости данного метода.

Работа выполнена при поддержке Лаборатории структурных методов анализа данных в предсказательном моделировании, МФТИ, грант правительства РФ дог. 11.G34.31.0073.

Список литературы

[1] Forrester A., Sobester A., Keane A. Engineering Design via Surrogate Modelling. A Practical Guide. *Wiley 2008 p. 238.*.

[2] Hastie T., Tibshirani R., Friedman J. he elements of statistical learning: data mining, inference, and prediction *Springer, 2008 p. 763*, .

[3] Беляев М.Г., Бурнаев Е.В., Любин А.Д. Методика формирования функционального словаря в задаче ап- проксимации многомерной зависимости *ММРО-15. с.146*, 2011.

[4] Беляев М, Любин А Особенности оптимизационной задачи, возникающей при построении аппроксимации многомерной зависимости *Труды научной конференции ИТИС - 2011.*

[5] Burnaev E. V., Belyaev M. G., Prihodko P. V. About hybrid algorithm for tuning of parameters in approximation based on linear expansions in parametric functions // Proceedings of the Intellectualization of information processing conference *IIP-2010, Moscow, 2010* .

[6] A. Pinkus Approximation theory of the MLP model in neural networks *Acta Numerica (8), Cambridge Univ. Press, Cambridge, 1999, pp. 143–195.*

[7] F. Scarselli, A. C. Tsoi Universal Approximation Using Feedforward Neural Networks: A Survey of Some Existing Methods,and Some New Results *Neural Networks. Elsevier, 1998. vol. 11, No. 1, pp. 15–37.*

[8] Nocedal J., Wright S. Numerical Optimization, 2nd Edition *Springer, 2006. — 664 p.*

[9] Dolan E., More J. enchmarking optimization software with performance profiles *Mathematical Programmming, Ser. A 91, 2002 pp. 201-213.*

[10] Burnaev E.V., Grihon S. Construction of the metamodels in support of stiffened panel optimization *Proceedings of the conference MMR 2009 – Mathematical Methods in Reliability, 22-29 June, pp. 124-128.*

[11] GDR MASCOT-NUM Toy Functions benchmark. `http://gdr-mascotnum.math.cnrs.fr/data2/benchmarks/jm.pdf`