Последовательное порождение существенно нелинейных моделей в задачах ранжирования документов*

Гущин А. Е.
1aguschin@gmail.com
Московский физико-технический институт

В работе исследуется алгоритм поиска функций ранжирования, являющихся суперпозициями элементарных математических функций. Обычно все модели, использующиеся для ранжирования документов, имеют две независимые переменные — частоту терма и частоту документа. В работе предлагается осуществить генерацию функций от этих переменных с помощью генетического программирования. Полученные функции протестированы на различных тестовых коллекциях. Приводится сравнение эффективности ранжирования функций, полученных в данной работе, с классическими функциями.

Ключевые слова: *информационный поиск, ранжирование документов, генетическое программирование.*

Введение

Задача ранжирования документов является одной из основных задач информационного поиска. Она заключается в поиске функции, наилучшим образом определяющей порядок документов в коллекции соответственно некоторому запросу.

С начала исследования информационного поиска разработано множество методов решения этой задачи, таких как vector space model [1], the language model [2], BM25 [3], и, более поздних, таких как HMM [4], DFR [5] и information-based models [6]. Все вышеперечисленные методы были разработаны в соответствии с теоретическими представлениями в области информационного поиска. Однако, так как не все методы легко объяснимы теоретически, при таком подходе к решению задачи ранжирования документов существует риск упустить некоторые более эффективные ранжирующие функции.

Ограничения этого подхода привели к исследованию пространства ранжирующих функций более систематически, несмотря на то, что такой подход ограничен сложностью пространства (его бесконечной размерностью и тем, что оно потенциально содержит все реальные функции). Первые попытки были основаны на использовании генетического программирования и генетических алгоритмов, для того, чтобы исследовать пространство функций ранжирования стохастически [12,16,6]. В работе [9] генетическое программирование было использовано для того, чтобы отыскать как оптимальные ранжирующие функции для отдельных запросов, так и одну оптимальную функцию для нескольких запросов. Однако, такой подход не лишен недостатков. Решения, сгенерированные данными алгоритмами часто сложно анализировать. Кроме того, генетическому программированию свойственно "раздувание кода когда практически все генетические алгоритмы имеют тенденцию производить всё более и более сложные функции с течением итераций. Таким образом, существует риск пропустить простые функции высокого качества.

Позднее исследователи сосредоточились на нахождении функций ранжирования как линейных комбинаций уже проверенных функций, параметры которых настраиваются с помощью тренировочных коллекций. В соответствии с этим методом были предложены

Численные методы обучения по прецедентам (практика, В.В. Стрижов)/Группа 174, весна 2014

различные методы "обучения ранжированию"[7] : поточечный подход (например, [8]), попарный подход (например, [9, 10, 11]), или списочный подход (например, [12]).

Несмотря на свою эффективность, последние методы имеют два ограничения: во-первых, обычно они представляют функцию ранжирования как функцию строго определенного вида (например, линейную или полиномиальную), во-вторых, им требуется тестовый набор данных для того, чтобы настроить параметры модели.

Наконец, в работе [5] предлагается решение описанных недостатков методом построения суперпозиций с помощью грамматики и словаря базисных функций. Однако, в связи со вычислительной сложностью генерации большого количества суперпозиций авторам пришлось ограничиться суперпозициями, состоящих менее чем из восьми функций и переменных.

В настоящей работе для порождения ранжирующих функций предлагается использование библиотеки индуктивного порождения нелинейных моделей Multivariate Regression Composer (MVR) для Matlab. Использованием генетического программирования предполагается исследовать функции, которые невозможно было проверить в работе [5]. С другой стороны, возникновение функций избыточной сложности, связанное с применением генетического программирования в MVR предлагается решить с помощью добавления штрафа за сложность суперпозиции. Генетическое программирование также не накладывает ограничений на вид возможных функций ранжирования.

Постановка задачи

Пусть дана выборка:

$$\mathfrak{S} = \{\mathfrak{D}_j | j = \{1, \ldots, K\}\},$$

$$\mathfrak{D}_j = \{(\mathbf{x}_i, y_i) | i = \{1, \ldots, N_j\},\ \mathbf{x}_i \in \mathbb{X} \subset \mathbb{R}^n,\ y_i \in \mathbb{Y} \subset \mathbb{R}\},$$

где K - число выборок; N_j - число элементов в выборке j; $\mathbf{x}_i$ - i-ый элемент выборки, обладающий n признаками; $\mathbb{X}$ - пространство значений вектора признаков, лежащее в $\mathbb{R}^n$; $\mathbb{Y}$ - множество значений зависимой переменной.

Требуется найти оптимальные в смысле максимизации некоторого функционала качества параметрические функции $f : \Omega \times \mathbb{X} \longmapsto \mathbb{R}$ вида

$$f(\mathbf{x}) = \sum_{q=1}^{n} \omega_q\ x_q + \sum_{q=n+1}^{n+m} \omega_q \varphi_q(\mathbf{x}),\ f(\mathbf{x}) \in \mathcal{H} \tag{1}$$

где $\mathbf{w} \in \Omega$ — вектор параметров модели, а функции $\varphi_q(\mathbf{x}) \in \mathcal{F}$ порождаются как суперпозиция базисных функций $g \in \mathcal{G}$, то есть как нелинейная суперпозиция признаков. Порождаемую суперпозицию можно представить как ориентированное дерево (связанный ациклический граф), вершина V_i которого представляет собой функцию $g_i \in \mathcal{G}$, где число дочерних вершин V_i равно арности функции g_i, а лист V_j представляет признак x_j либо числовой параметр w_j.

Порождаемая суперпозиция должна быть допустимой, то есть суперпозиция $\varphi_q(\mathbf{x})$ должна быть определена для любых $\mathbf{x} \in \mathbb{R}^n$. Таким образом, для функции $g(g_1(\mathbf{x}), ..., g_k(\mathbf{x}))$ область значений функций $g_1(\mathbf{x}), ..., g_k(\mathbf{x})$ должна быть подмножеством области допустимых значений функции $g(x_1, ..., x_k)$.

Элементы множества $\mathcal{F} = \{\varphi_q(\mathbf{x})\}$ могут быть порождены индуктивно как суперпозиции функций из некоторого заданного набора элементарных функций $\mathcal{G} = \{g_1, \ldots, g_m\}$.

Таким образом, задача состоит в том, чтоб найти функции $f_{\hat{r}} \in \mathcal{H}$, имеющие максимальное значение некоторого функционала качества Q:

$$\hat{r} = \arg\max_{r \in \mathbb{N}}(Q(f_r|\hat{\boldsymbol{\omega}}_r, \mathfrak{D})), \tag{2}$$

где $\hat{\boldsymbol{\omega}}_r$ - оптимальный набор параметров модели f_r:

$$\hat{\boldsymbol{\omega}}_r = \arg\max_{\boldsymbol{\omega} \in \Omega}(Q(\boldsymbol{\omega}|\ f_r, \mathfrak{D})), \tag{3}$$

Рассмотрим подробнее используемый функционал качества Q. Пусть некоторая коллекция документов $\{x_i\} \in \mathbb{X}$ была ранжирована в соответствии с некоторой функцией и было выбрано $\mathbf{p}$ наиболее релевантных документов. Эти $\mathbf{p}$ документов были также оценены экспертом, который выставил для i-того документа оценку y_i от 0 до 3: нерелевантный документ - 0, полностью релевантный - 3, для промежуточных оценок - 1 и 2. Для оценки качества ранжирующей функции воспользуемся DCG (Discounted cumulative gain):

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{y_i} - 1}{\log_2(i+1)}, \tag{4}$$

Количество выбранных документов $\mathbf{p}$ может отличаться для различных запросов. Для того, чтобы оценить ранжирующую функцию на различных запросах, необходимо нормализовать DCG. Для этого обозначим $\mathrm{IDCG_p}$ значение, которое принимает $\mathrm{DCG_p}$ на идеально ранжированной выборке. Тогда нормализованная DCG:

$$\mathrm{nDCG_p} = \frac{\mathrm{DCG_p}}{\mathrm{IDCG_p}}, \tag{5}$$

Алгоритм решения задачи ранжирования с выбором нелинейных признаков

Выбор признаков среди $\{x_1, ..., x_n, \varphi_1, ..., \varphi_m\}$ и оценка параметра $\mathbf{w}$

Выбор наилучшей модели осуществим относительно среднеквадратичной ошибки

$$S(\mathbf{w}) = \sum_{i=1}^{k}(y_i - f(\mathbf{w}, \mathbf{x}_i))^2,$$

проведя для этого отбор наиболее информативных признаков. Искомая модель имеет вид $f^*(\mathbf{x}) = \varphi(\mathbf{w}^*, \mathbf{x})$, где параметры модели $\mathbf{w}^*$ должны доставлять минимум функции:

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^n} S(\mathbf{w}|X^\ell, \varphi_F),$$

где X^ℓ — обучающая выборка. Таким образом, необходимо определить множество $F \subseteq I$, где $I = \{x_1, ..., x_n, \varphi_1, ..., \varphi_m\}$, которое доставляло бы минимум функции:

$$F^* = \arg\min_{F \subseteq I} S(\varphi_F|\mathbf{w}^*, X^C),$$

где X^C — контрольная выборка, φ_F — модель φ, включающая только столбцы из множества F.

Отбор признаков предлагается провести методом последовательного добавления признаков, и последующего поиска F^*.

На этапе добавления очередного признака находим признак i^*:

$$i^* = \arg\min_{i \in I \setminus F_{k-1}} S(\mathbf{w}|X^\ell, \varphi_{F_{k-1} \cup \{i\}})$$

и добавляем найденный признак $F_k = F_{k-1} \cup i^*$.

После окончания работы алгоритма (то есть после включения всех признаков в модель), находим искомое множество F^* среди $\{F_i\}_{i \in I}$ и соответствующую этому множеству модель $f^*(\mathbf{x})$.

Порождение нового признака φ_q

Алгоритм порождения нового признака φ_q:

1. Для порождения нового признака воспользуемся множеством признаков F^*. Построим функционал качества (5) на обучающей выборке X^ℓ.
2. Сгенерируем случайные суперпозиции $\{\varphi_q(X)\}$ и добавим их к уже имеющимся.
3. Произведем скрещивание и мутацию моделей $\{\varphi_q(X)\}$.
   (а) Скрещивание двух моделей происходит с помощью обмена случайно выбранного поддеревьева первой модели на случайно выбранное поддерево второй модели.
   (б) Мутация моделей происходит путём замены случайно выбранной вершины дерева модели на новую случайно сгенерированную модель.
4. Настроим параметры ω_q моделей $\{\varphi_q(X)\}$ на X^ℓ.
5. Построим функционал качества (5) на контрольной выборке X^C. Выберем фиксированное количество моделей, имеющих наибольшее значение функционала качества на X^C.
6. Если не достигнуто ни требуемое качество на X^C, ни максимальное число итераций, перейдем на следующую итерацию к пункту 2.
7. Выберем признак, имеющий наилучшее качество на X^C.

Вычислительный эксперимент

Целью вычислительного эксперимента является генерация функций ранжирования и проверка их качества. Функция ранжирования коллекции документов является суперпозицией элементарных унарных (логарифм, экспонента, квадратный корень) и бинарных (сложение, вычитание, умножение, деление, степенная функция) функций.

В качестве средства генерации была взята библиотека MVR

(http://sourceforge.net/projects/mvr/), использующая генетические алгоритмы. В качестве функционала качества была взята nDCG. Проверка качества осуществлялась при нахождении nDCG на тестовой выборке, не использовавшейся на этапе генерации функций.

В качестве данных используются реальные таблицы оценок, предоставленные Яндексом (http://imat2009.yandex.ru/academic/mathematic/2009/datasets). Таблицы содержат уже посчитанные и нормализованные признаки пар «запрос-документ», а также оценки релевантности, сделанные асессорами (оценщиками качества поиска) Яндекса. Таблицы не содержат оригинальных запросов и ссылок на оригинальные документы, не описана семантика признаков (признаки просто пронумерованы). Примеры признаков, участвующих в таблице, – частота слова в документе, частота документов с данным словом в коллекции (tf*idf), длина запроса в словах.

Для проведения вычислительного эксперимента было выбрано 4 информативных признака.

Каждая строка файлов данных соответствует паре «запрос-документ». Все признаки либо бинарные – принимают значения из 0, 1, либо непрерывные. Значения непрерывных признаков нормированы на интервал [0, 1]. Каждой паре «запрос-документ» соответствуют значения 245 признаков. Если значение признака равно 0, то он опускается. В комментариях в конце каждой строки указан идентификатор запроса. Файл с обучающей выборкой содержит оценку релевантности, значения из диапазона [0, 4] (4 – «высокая релевантность», 0 – «нерелевантно»).

При нахождении оптимальных параметров моделей максимизировалась суммарная nDCG для всех запросов $\mathbf{Q}$ в обучающей выборке:

$$\mathrm{nDCG}_{\mathrm{p}}^{\mathrm{sum}} = \sum_{\mathrm{q}\in\mathrm{Q}} \frac{\mathrm{DCG}_{\mathrm{p}}}{\mathrm{IDCG}_{\mathrm{p}}}, \tag{6}$$

Для контроля качества находилась средняя nDCG для тестовой выборки.

Далее под линейной моделью понимается функция $f(w, x) = w * x^T$, где параметры w подбираются так, чтобы уменьшить среднеквадратичную ошибку на обучающей выборке. Соответственно линейная модель не сравнивает зависимые переменные как модели, генерируемые mvr.sl, а предсказывает оценку ассессора.

На данном этапе вычислительного эксперимента предлагается сгенерировать с помощью mvr.sl несколько признаков, затем добавить их к уже имеющимся и оценить изменение качества линейной модели на тестовой выборке.

Для начала воспользуемся синтетическими данными. Для этого воспользуемся функцией Розенброка:

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} \left[(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2\right] \quad \forall, x \in \mathbb{R}^N,$$

при $N = 40$. Добавим к значениям $f(\mathbf{x})$ шум, равный 1% от среднего значения $f(\mathbf{x})$, имеющий нормальное распределение. Добавим 10 шумовых переменных, имеющих нормальное распределение.

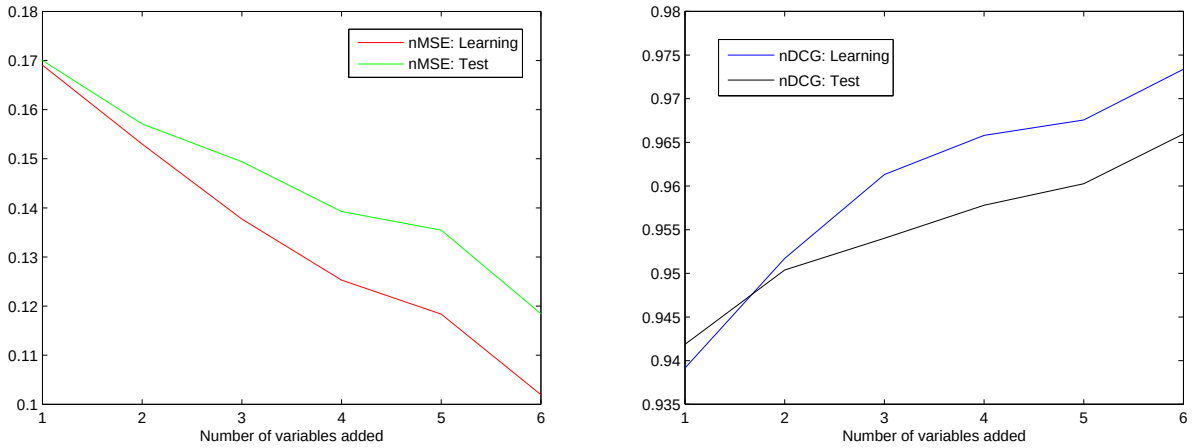
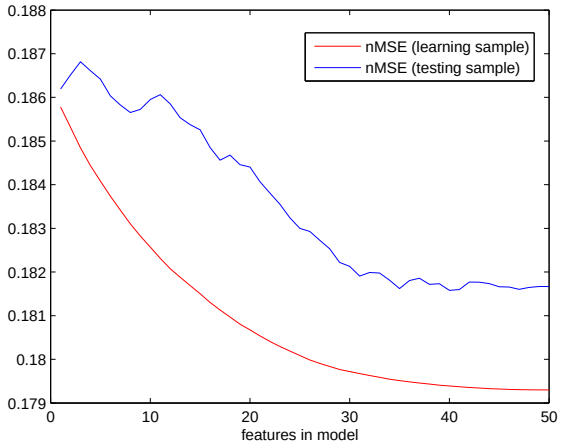
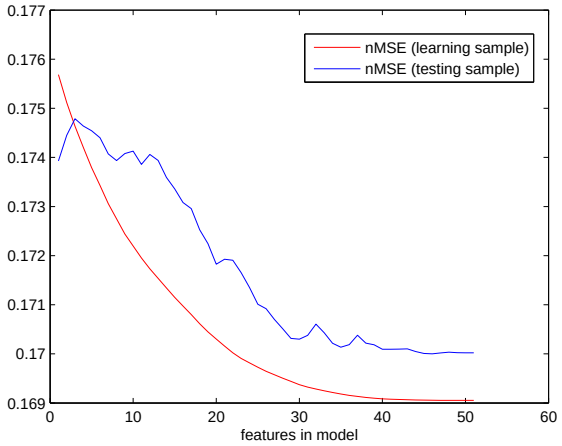


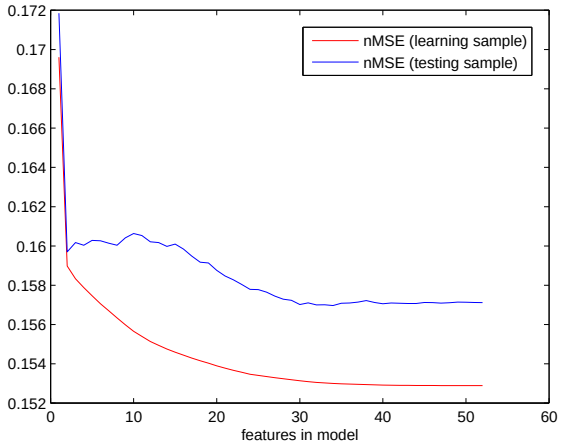
Рис. 1. Изменение качества линейной модели при пошаговом добавление сгенерированных переменных.



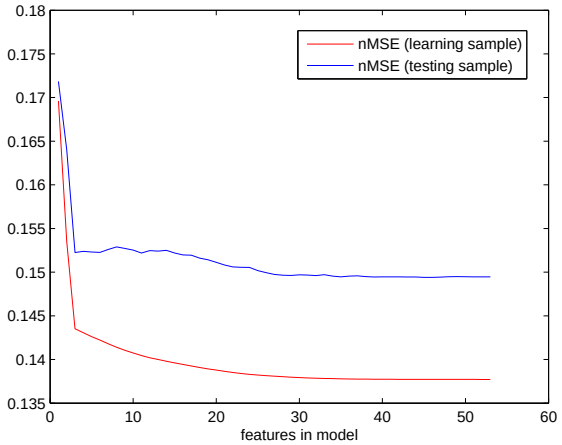
(a) 50 исходных признаков



(b) добавлен один сгенерированный признак



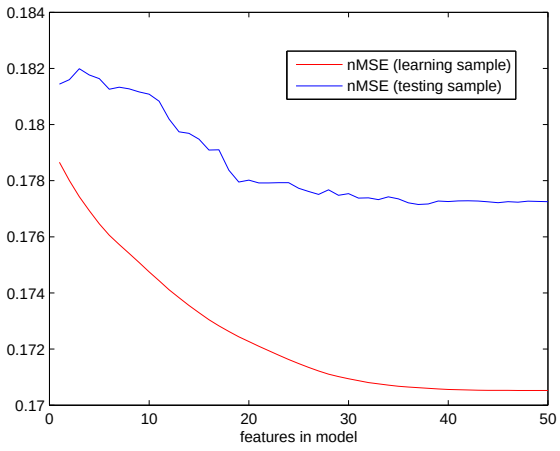
(c) добавлено два сгенерированных признака



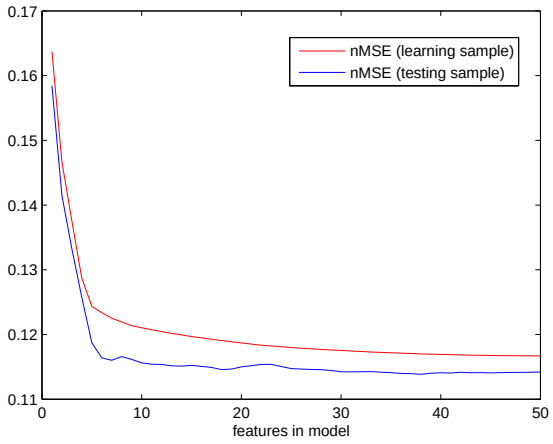
(d) добавлено три сгенерированных признака

Рис. 2. Пошаговое добавление переменных при поиске линейной модели.

На графике ниже изображена нормированная среднеквадратичная ошибка для линейной модели в зависимости от числа использованных переменных (переменные добавлялись по одной таким образом, чтобы получившаяся модель давала наилучшее качество на обучающей выборке). Минимум ошибки на тестовой выборке достигается при 40 переменных и равен 0.1676. Эти переменные были использованы для порождения новых признаков. После добавления к имеющимся признакам 8 сгенерированных признаков, о которых написано ниже, минимум ошибки на тестовой выборке достигается при 38 переменных и равен 0.1139.



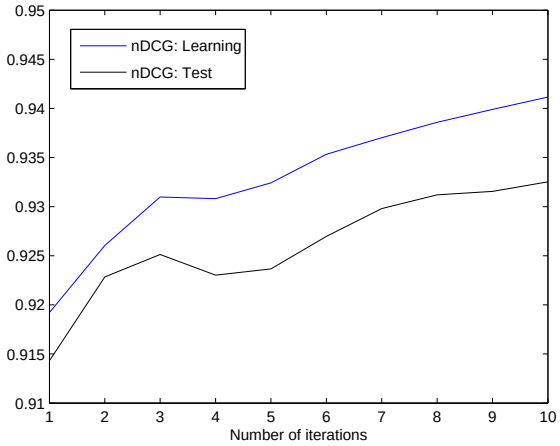
(a) 50 исходных признаков



(b) дополнительно 8 сгенерированных признаков

Рис. 3. Зависимость ошибки от количества признаков в линейной модели на обучающей и тестовой выборках.

С помощью mvr.sl были сгенерированы 8 признаков и добавлены к имеющимся 50. Два графика, описывающие качество и сложность генерируемых признаков приведены ниже.



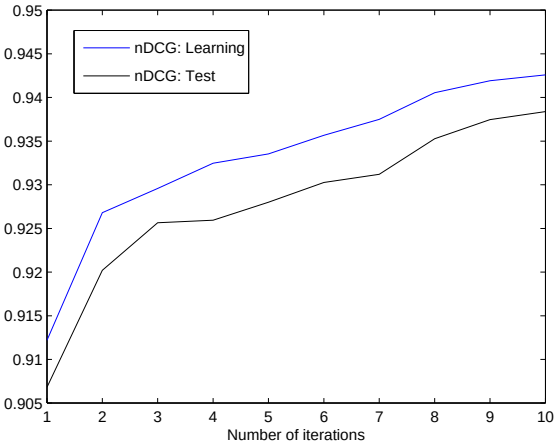
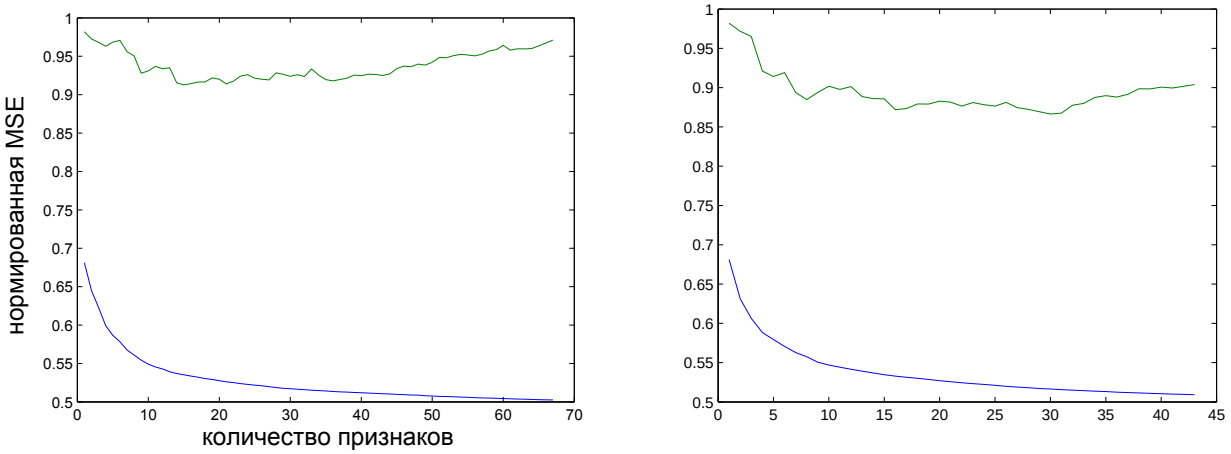


Рис. 4. Примеры порождаемых mvr.sl признаков.

Для линейной модели из 40 переменных, настраиваемой на обучающей выборке без порожденных признаков, nDCG на тестовой выборке составило 0.9329, а для линейной модели из 38 переменных, настраиваемой на обучающей выборке с порожденными признаками, nDCG составило 0.9618.

Теперь воспользуемся реальными данными Яндекса. На графике ниже изображена нормированная среднеквадратичная ошибка для линейной модели в зависимости от числа использованных переменных (переменные добавлялись по одной таким образом, чтобы получившаяся модель давала наилучшее качество на обучающей выборке). Минимум ошиб-

ки на тестовой выборке достигается при 15 переменных и равен 0.9129. Эти переменные были использованы для порождения новых признаков. После добавления к имеющимся признакам 7 сгенерированных признаков, о которых написано ниже, минимум ошибки на тестовой выборке достигается при 30 переменных и равен 0.8664.



(a) 245 исходных признаков

(b) дополнительно 7 сгенерированных признаков

Рис. 5. Зависимость ошибки от количества признаков в линейной модели на обучающей и тестовой выборках. Синим цветом изображено качество на обучающей выборке, зеленым - на тестовой.

С помощью mvr.sl были сгенерированы 7 признаков и добавлены к имеющимся 245. Два графика, описывающие качество и сложность генерируемых признаков приведены ниже.

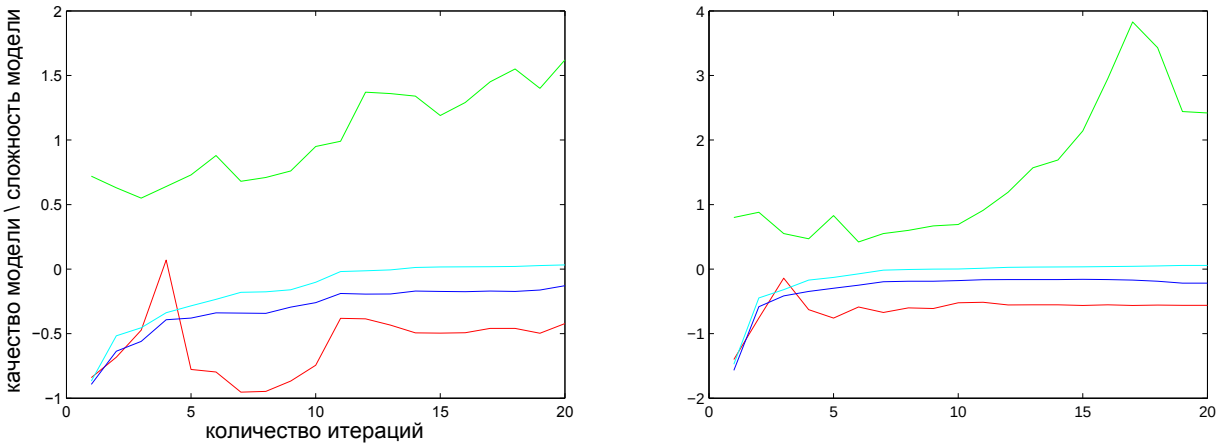


Рис. 6. Примеры порождаемых mvr.sl признаков. Зеленой линией обозначена сложность модели (число элементарных функций и признаков), деленная на 10. Качество моделей измеряется в nDCG. Нулю на вертикальной оси соответствует значение nDCG для линейной модели из 15 признаков, выбранных при построении линейной модели, единице соответствует nDCG=1. Красным цветом обозначены качество на 100 элементах выборки, использующихся для построения и максимизации nDCG, бирюзовым цветом обозначено качество на 3900 элементах выборки, использующееся для выбора лучших моделей для следующей итерации, синим цветом обозначено качество моделей на 4000 элементах тестовой выборки.

Для линейной модели из 15 переменных, настраиваемой на обучающей выборке без порожденных признаков, nDCG на тестовой выборке составило 0.8778, а для линейной модели из 30 переменных, настраиваемой на обучающей выборке с порожденными признаками, nDCG составило 0.8804.

Формат реальных данных

Функции ранжирования присваивают документу $\mathbf{d}$ некоторое положительное значение в ответ на запрос $\mathbf{q}$, состоящий из термов $\mathbf{w}$. Классически в ранжировании документов используются следующие переменные: частота терма $t^{\mathbf{d}}_{\mathbf{w}}$ и частота документа $N_{\mathbf{w}}$.

Известно, что нормализованные величины более эффективны в задачах ранжирования документов. Например, подобные относительные величины используются в языковой модели [3], в BM25 используется нормализация Okapi [13]. Для этой работы были выбраны величины, использующиеся в DFR и information-based models [6], а также в работе [5].

— Нормализованная частота терма $x^{\mathbf{d}}_{\mathbf{w}} = t^{\mathbf{d}}_{\mathbf{w}} log(1 + c\frac{l_{avg}}{l_{\mathbf{d}}})$, где $c \in R$ — множитель. Для простоты далее вместо $x^{\mathbf{d}}_{\mathbf{w}}$ будем писать x
— Нормализованная частота документа $y_{\mathbf{w}} = \frac{N_{\mathbf{w}}}{N}$ (нужны ли объяснения?). Для простоты далее вместо $y_{\mathbf{w}}$ будем писать y
— Константа $k \in R$

Литература

[1] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1983.

[2] Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

Таблица 1. Использующиеся обозначения

Обозначение	Определение
$t^{\mathbf{d}}_{\mathbf{w}}$	Количество появлений терма $\mathbf{w}$ в документе $\mathbf{d}$, частота терма
$t^{\mathbf{q}}_{\mathbf{w}}$	Количество появлений терма $\mathbf{w}$ в запросе $\mathbf{q}$
$x^{\mathbf{d}}_{\mathbf{w}}$	Нормализованная частота терма
$N_{\mathbf{w}}$	Количество документов в коллекции, содержащих $\mathbf{w}$, частота документа
$y_{\mathbf{w}}$	Нормализованная частота документа
N	Количество документов в коллекции
$l_{\mathbf{d}}$	Длина документа $\mathbf{d}$ в количестве термов
l_{avg}	Средняя длина документа в коллекции

[3] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49. ACM, 2004.

[4] Donald Metzler and W Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

[5] Gaussier E. Amini M.R. Goswami P., Moura1 S. Exploring the space of ir functions. 2014.

[6] Stéphane Clinchant and Eric Gaussier. Information-based models for ad hoc ir. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 234–241. ACM, 2010.

[7] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[8] Koby Crammer, Yoram Singer, et al. Pranking with ranking. In *NIPS*, volume 14, pages 641–647, 2001.

[9] William W Cohen, Robert E Schapire, and Yoram Singer. Learning to order things. *arXiv preprint arXiv:1105.5464*, 2011.

[10] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[11] Thorsten Joachims. Making large scale svm learning practical. 1999.

[12] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing ndcg measure. In *NIPS*, volume 22, pages 1883–1891, 2009.

[13] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.