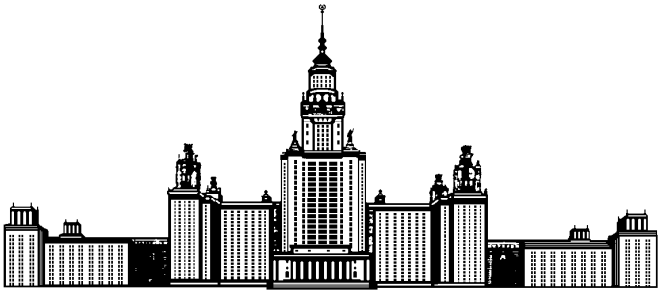


Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра Математических методов прогнозирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Применение метода распространения ожидания для решения задачи прогнозирования

Выполнил:
студент 417 группы
Чистяков Александр Сергеевич

Научный руководитель:
к.ф.-м.н., доцент
Ветров Дмитрий Петрович

Москва, 2015

Содержание

1 Введение

Данная работа посвящёна разработке вероятностной модели для прогнозирования результатов футбольных матчей. Задача прогнозирования результатов спортивных соревнований является важной, так как имеет большую практическую значимость в сфере развития букмейкерских компаний и при разработке регламента различных массовых состязаний. При этом данная задача прогнозирования является сложной, потому что требует учёта нетривиальной структуры прогнозируемых состязаний, а также формального описания возникающих естественным образом случайных процессов, влияющих на исход состязаний.

Одним из способов предсказания результатов матчей является построение вероятностной модели матча. В частности, в настоящее время существуют вероятностные модели сеории TrueSkill. После построения модели, возникают две главные задачи: обучение параметров модели и предсказание исхода матча. Одним из самых успешных методов для этих задач является метод распространения ожидания (Expectation Propagation, EP), являющийся обобщением известного алгоритма Loopy Belief Propagation на случай непрерывных случайных величин и сложных метрик для аппроксимации распределений.

Большинство вероятностных моделей исхода матчей были разработаны для предсказания результатов шахматных партий, поэтому, при применении к футбольным матчам, не могут учесть важную вспомогательную информацию: например, счёт матча.

В рамках данной работы разработана новая вероятностная модель, которая учитывает дополнительные особенности соревнований, присущие футбольным матчам: счёт матча и влияние игры на домашнем стадионе на результативность команды. При помощью библиотеки Infer.NET, реализующей метод Expectation Propagation, данная вероятностная модель была реализована и протестирована на реальных данных футбольных турниров. Экспериментальное сравнение показывает, что учёт дополнительной информации действительно улучшает качество прогноза исхода матчей.

Работа структурирована следующим образом:

- В секции 2 работы продемонстрированы основные идеи, используемые при построении метода распространения ожидания. Также в этой секиции описывается процесс сведения построения аппроксимаций к применению итерационной схемы пересчёта сообщений;
- В секции 3 осуществляется сравнение различных метрик, используемых при простроении аппроксимации, и приводятся существующие в настоящее время теоретические результаты, позволяющие удобно осуществлять аппроксимацию при работе с распределениями из экспоненциального семейства;
- Секция 4 посвящена описанию возможностей библиотеки Infer.NET, позволяющей эффективно вести разработку вероятностных моделей и осуществлять вывод в них с помощью различных алгоритмов, использующих схему передачи сообщений;
- В секции 5 приводится описание существующих в настоящее время вероятностных математических моделей, позволяющих осуществлять прогнозирование результатов различных соревнований;
- В секции 6 приводятся результаты экспериментов, направленных на прогнозирование результатов шахматных турниров с помощью модели TTT-D;
- Секция 7 посвящена описанию недостатков существующих моделей, прогнозирующих результаты шахматных партий, при использовании их для прогнозирования результатов футбольных матчей. В этой же секции приводится описание разработанной вероятностной модели FootballSkill, учитывающей специфичную для футбольных турниров информацию;
- В секции 8 приводятся результаты экспериментов с разработанной моделью на реальных данных футбольных матчей России и СССР; проводится сравнение модели FottballSkill с существующими аналогами.

2 Алгоритм распространения ожидания

2.1 Модель с одной скрытой переменной

При построении вероятностных моделей в машинном обучении часто оказывается, что для вычисления некоторых распределений требуется чрезмерно большое количество арифметических операций и осуществить их за примелимое время не представляется возможным. Однако во многих случаях искомое распределение удаётся

аппроксимировать, приблизив его более простым, и решить поставленную задачу, затратив гораздо меньше времени.

Рассмотрим один из таких методов аппроксимации на примере задачи обнаружения источника сигнала. Простейшая постановка задачи звучит следующим образом: пусть на вещественной оси в точке с координатой x расположен источник. Имеется датчик, который регистрирует сигналы от источника. Известно, что координаты точки обнаружения сигнала y_i зашумлены и имеют нормальное распределение с центром в точке x. Помимо сигналов от источка имеется общий шумовой фон из-за которого иногда датчик обнаруживает лишние сигналы. Таким образом, плотность распределения точек y_i имеет вид:

$$p(y_i|x) = \alpha \cdot \mathcal{N}(y_i; x, 1) + (1 - \alpha) \cdot \mathcal{N}(y_i; 0, 10) \tag{1}$$

Задача состоит в определении расположения источника x по известным координатам зарегистрированных сигналов y_i. Чтобы вероятностная постановка задачи была корректной, в задаче предполагается, что существует априорное распределение $p(x)$ на координату источника.

Пользуясь формулой Байеса и независимостью зарегистрированных сигналов, можно получить постериорное распределение на координату x:

$$p(x|y_1, y_2, \ldots, y_N) \propto p(x) \cdot p(y_1|x) \cdot p(y_2|x) \cdot \ldots \cdot p(y_N|x) \tag{2}$$

Будем считать, что априорное распределение представляет собой гауссиану с нулевым математическим ожиданием и большой дисперсией:

$$p(x) = \mathcal{N}(x; 0, 100) \tag{3}$$

В этом случае, выражение в формуле (2) после раскрытия скобок будет представлять собой сместь 2^N гауссиан. Заметим, что условное распределение на x известно с точностью до нормировочной константы и, если количество точек y_i достаточно велико, то вычислить эту константу за разумное время оказывается невозможным. Также оказывается что невозможно эффективно найти моду этого распределения для построения точечной оценки на координату источника.

Условное распределение на x можно записать в виде произведения функций-факторов:

$$p(x|y_1, y_2, \ldots, y_N) \propto \prod_{i=0}^{N} f_i(x) \tag{4}$$

где f_0 соответствует априорному распределению $p(x)$, а остальные множители f_i условным распределениям $p(y_i|x)$. Схематически такие распределения изображают в виде фактор-графа (см. Рис. 1)

Прямоугольниками в фактор-графе обозначаются сами функции-факторы, а кругами — переменные, от которых эти факторы зависят. Известно, что если структура фактор-графа представляет собой дерево, и все переменные имеют дискретное распределение на конечном множестве, то вычисление всех маргинальных распределений, а также нормировочной константы в совместном распределении можно эффективно осуществить с помощью алгоритма распространения доверия (Loopy Belief Propagation, LBP) [8].

В некоторых случаях непрерывные распределения можно приблизить дискретными, поделив носитель распределения на конечное число интервалов. Однако в данной задаче такой подход не даст хороших результатов. Чтобы построить построить аппроксимацию приблизим каждый фактор $f_i(x)$ функцией $\hat{f}_i(x)$ из некоторого семейства $\mathcal{F}$:

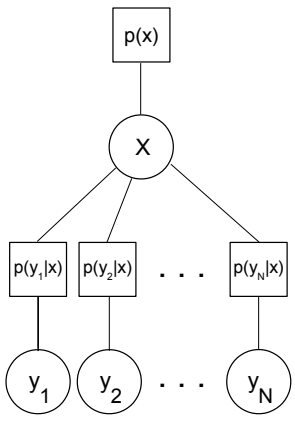


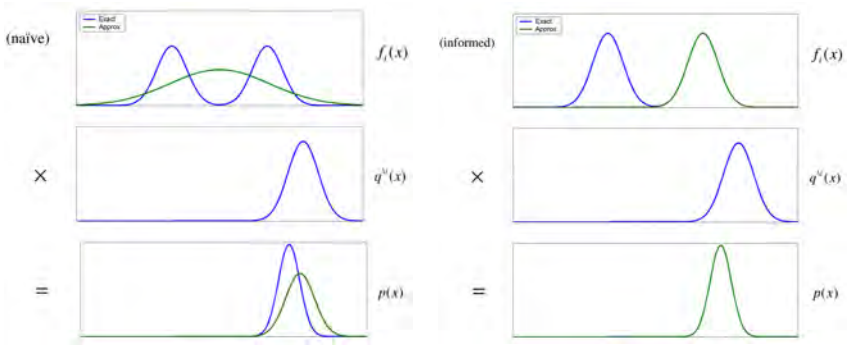
Рис. 1. Фактор-граф к задаче поиска источника сигнала

$$p(x|Y) \propto \prod_{i=0}^{N} f_i(x) \approx \prod_{i=0}^{N} \hat{f}_i(x), \quad \hat{f}_i \in \mathcal{F} \tag{5}$$

Главная идея аппроксимации состоит в том, что приближать факторы нужно не по отдельности, а с учётом контекста, задаваемого остальными множителями. То есть:

$$\prod_{i=0}^{N} f_i(x) \approx \prod_{i=0}^{N} \hat{f}_i(x) \qquad \not\Leftarrow \qquad \{f_i(x) \approx \hat{f}_i(x)\}_{i=0}^{N}$$

Чтобы убедиться в этом рассмотрим пример, представленный на (Рис. 2).



(a) $f_i(x) \approx \hat{f}_i(x)$ (b) $f_i(x) \cdot q^{\setminus i}(x) \approx \hat{f}_i(x) \cdot q^{\setminus i}(x)$

Рис. 2. Сравнение двух подходов к аппроксимации

Слева (Рис. 2a) бимодальная функция $f_i(x)$ приближается одной гауссианой. Функция $q^{\setminus i}(x) = \prod_{k \neq i} \hat{f}_k(x)$ описывает контекст, задаваемый остальными факторами $f_k(x)$. Результаты перемножения исходной функции и её аппроксимации на произведение остальных факторов оказываются явно различными. Справа (Рис. 2b) при построении аппроксимации учтено, что при всех x, соответствующих левой моде аппроксимируемой функции, $q^{\setminus i}(x)$ практически равно 0, а следовательно имеет смысл приближать гауссианой только правую моду.

Учитывая данное наблюдение, построение аппроксимации распределения $p(x|Y)$ можно свести к следующему алгоритму:

1. В качестве базового семейства $\mathcal{F}$ зафиксируем пространство одномерных гауссиан.
2. Зафиксируем все аппроксимирующие функции $f_j(x)$ кроме i-той и обозначим $q^{\setminus i}(x) = \prod_{k \neq i} \hat{f}_k(x)$. По свойству нормального распределения функция $q^{\setminus i}(x)$ также будет являться гауссианой.

   Такми образом

   $f_i(x)$ — произвольная функция;

   $\hat{f}_i(x)$ — гауссиана;

   $q^{\setminus i}(x)$ — гауссиана
3. Обновим $\hat{f}_i(x)$, потребовав выполнения приближенного равенства:

$$\hat{f}_i(x) \cdot q^{\setminus i}(x) \approx f_i(x) \cdot q^{\setminus i}(x) \tag{6}$$

   Слева в результате произведения получается гауссиана. Справа — смесь двух гауссиан. Введём операцию проецирования на базовое семейство аппроксимации:

$$\text{proj}[p(x)] = q(x): \quad q \in \mathcal{F}, q(x) \approx p(x) \tag{7}$$

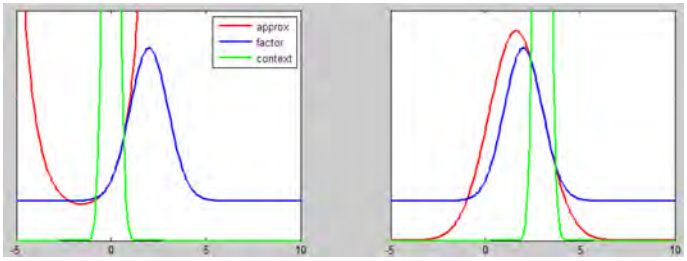
   В данной ситуации можно считать, что проецирование осуществляется путём приравнивания первого и второго момента у гауссианы и аппроксимируемого распределения. Как будет показано далее, это далеко не единственный способ определить операцию проецирования, но в рассматриваемой задаче он работает достаточно хорошо. Воспользовавшись введённым оператором можно записать выражение

$$\hat{f}_i(x) \cdot q^{\setminus i}(x) = \text{proj}\left[\hat{f}_i(x) \cdot q^{\setminus i}(x)\right] \tag{8}$$

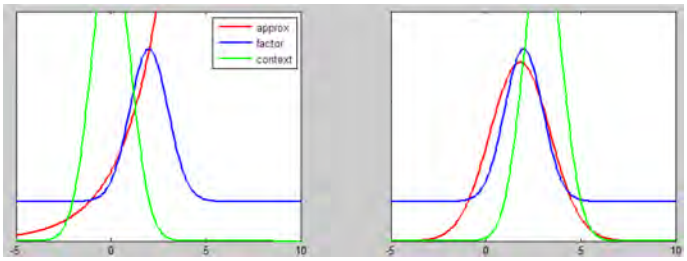
Из которого сразу получается формула для обновления аппроксимирующей функции

$$\hat{f}_i(x) = \frac{\text{proj}\left[\hat{f}_i(x) \cdot q^{\setminus i}(x)\right]}{q^{\setminus i}(x)} \tag{9}$$

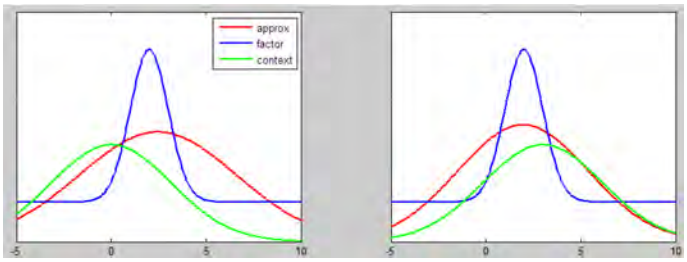
4. Повторяя итерационно процесс из пунктов 1-3 для различных факторов можно добиться сходимости алгоритма и получить искомую аппроксимацию на распределение $p(x|Y)$.



(a) Аппроксимация с учётом узкого контекста



(b) Аппроксимация с учётом среднего контекста



(c) Аппроксимация с учётом широкого контекста

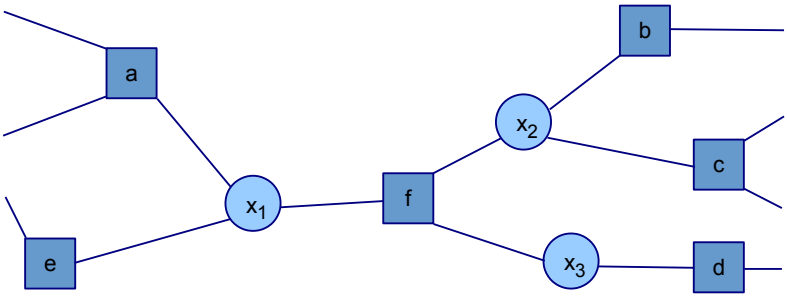
Рис. 3. Аппроксимация факторов в задаче обнаружения источника сигнала

Данный подход представляет собой простейшую версию реализации алгоритма Expectation Propagation для модели с одной переменной. На графиках 3a, 3b, 3c показан результат построения аппроксимации в рассмотренной задаче обнаружения источника сигнала. Видно, что ширина колокола у графика, соответствующего контекстному множителю $q^{\setminus i}(x)$, явно задаёт отрезок на котором построенная аппроксимация $\hat{f}_i(x)$ пытается приблизить исходный фактор $f_i(x)$. При чём оказывается, что при увеличении количества факторов, участвующих в аппроксимации, ширина учитываемого контекста сужается и аппроксимация приближаемого распределения становится всё точнее.

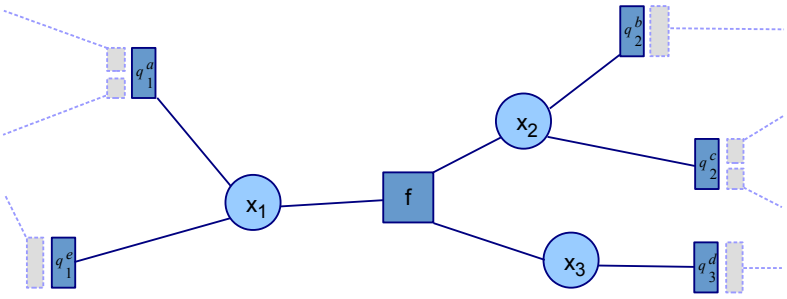
Вообще говоря не существует теоретических гарантий сходимости алгоритма по итерациям и получения хорошего результата, однако на практике метод работает и позволяет получить одно из самых точных решений для данной задачи [4].

2.2 Модель с несколькими скрытыми переменными

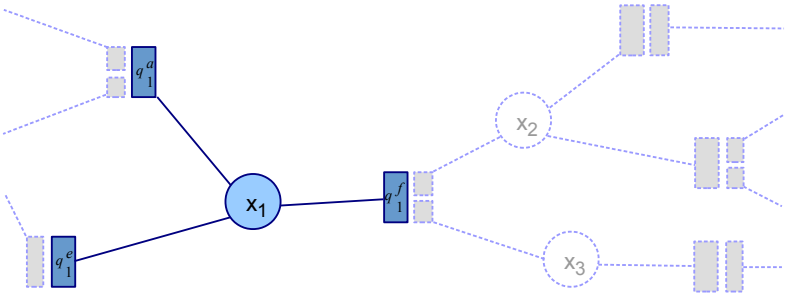
В рассмотренной задаче обнаружения источника сигнала использовалась всего одна скрытая переменная x. Рассмотрим как следует модифицировать данный метод, чтобы его можно было применять для более сложных графических моделей.



(a) Исходный фактор-граф



(b) Локализация контекста для аппроксимации фактора $f(x_1, x_2, x_3)$



(c) Построение аппроксимации $f(x_1, x_2, x_3) \approx q_1^f(x_1) q_2^f(x_2) q_3^f(x_3)$

Рис. 4. Аппроксимация фактор-графа со многими переменными

Как и в случае с одной переменной, сложное совместное распределение описывается через функции-факторы, зависящие от входящих в модель переменных. При этом считется, что фактор f_i зависит только от подмножества переменных X_i, с которым он соединён рёбрами.

$$p(X) \propto \prod_{i=1}^{N} f_i(X_i) \tag{10}$$

На (Рис. 4a) изображён фрагмент сложного фактор-графа. На представленном фрагменте фактор f зависит от подмножества переменных $X_f = \{x_1, x_2, x_3\}$.

Вообще говоря можно снова пытаться приблизить каждый фактор $f_i(X)$ более простой функцией из некоторого семейства $\hat{f}_i(X)$, однако теперь факторы имеют более сложную структуру, и хорошее базовое семейство, позволяющее эффективно перемножать, делить и проецировать факторы редко удаётся найти.

Упростим задачу и потребуем, чтобы каждый из аппроксимированных факторов был представим в виде произведения функций одной переменной из базового семейства:

$$p(X) \propto \prod_i f_i(X_i) \approx \prod_i \hat{f}_i(X_i) = \prod_i \left(\prod_{x_j \in X_i} q_j^{f_i}(x_j) \right) = \prod_j Q_j(x_j) \tag{11}$$

Сразу отметим, что до использования аппроксимации, вычисление маргинального распределения одной скрытой переменной было трудно осуществимо:

$$p(x_1) \propto \int \prod_i f_i(X_i) dX^{\setminus 1} \tag{12}$$

Для построения аппроксимации $\hat{f}(X_f)$ зафиксируем имеющиеся аппроксимации у всех остальных факторов. Поскольку в результате упрощения все прочие факторы раскладываются в произведение независимых функций, контекст, учитываемый при построении $\hat{f}$, тоже существенно упростится.

Рассмотрим процесс построения аппроксимации в полученном упрощённом фактор-графе (Рис. 4b). Для этого запишем маргинальное распределение на переменную x_1.

$$p_{[context]}(x_1) \propto \int\int f(x_1, x_2, x_3) \left[q_1^a(x_1) q_1^e(x_1)\right] \left[q_2^b(x_2) q_2^c(x_2)\right] \left[q_3^d(x_3)\right] dx_2 dx_3 =$$
$$= \int\int f(x_1, x_2, x_3) m_{x_1 \to f}(x_1) m_{x_2 \to f}(x_2) m_{x_3 \to f}(x_3) dx_2 dx_3 \tag{13}$$

Теперь, следуя логике алгоритма, полученного для одной переменной, нужно потребовать, чтобы при замене фактора f на его аппроксимацию распределение в модели изменилось как можно меньше.

Заменим фактор $f(x_1, x_2, x_3)$ произведением функций $q_1^f(x_1)$, $q_2^f(x_2)$ и $q_3^f(x_3)$ (Рис. 4c) и снова запишем полученное маргинальное распределение на переменную x_1.

$$p_{[approx]}(x_1) \propto q_1^f(x_1) [q_1^a(x_1) q_1^e(x_1)] = q_1^f(x_1) m_{x_1 \to f}(x_1) \tag{14}$$

Предполагая, что все функции $q_j^{f_i}$ принадлежат простому базовому семейству $\mathcal{F}$, позволяющему эффективно умножать, делить и осуществлять проецирование, из формул 13 и 14 можно получить формулу для пересчёта функции $q_1^f(x_1)$.

$$q_1^f(x_1) m_{x_1 \to f}(x_1) = \text{proj} \left[\int\int f(x_1, x_2, x_3) m_{x_1 \to f}(x_1) m_{x_2 \to f}(x_2) m_{x_3 \to f}(x_3) dx_2 dx_3 \right] \tag{15}$$

$$q_1^f(x_1) = \frac{\text{proj} \left[\int\int f(x_1, x_2, x_3) m_{x_1 \to f}(x_1) m_{x_2 \to f}(x_2) m_{x_3 \to f}(x_3) dx_2 dx_3 \right]}{m_{x_1 \to f}(x_1)} \tag{16}$$

Формулы для пересчёта остальных $q_j^{f_i}$ можно получить аналогично. В общем случае,

$$q_j^{f_i}(x_j) = \frac{\text{proj} \left[\int f(X_i) \prod_{x_k \in X_i} m_{x_k \to f_i}(x_k) dX_i^{\setminus j} \right]}{m_{x_j \to f_i}(x_j)} \tag{17}$$

У полученной формулы 17 есть важная особенность: если распределение, стоящее под опреатором проецирования окажется принадлежащим семейству $\mathcal{F}$ (например, это верно для случая дискретного распределения на всех переменных), то от проецирования можно будет избавиться, и множитель $m_{x_j \to f_i}(x_j)$ в числителе и знаменателе сократится. В этом случае формула для пересчёта $q_j^{f_i}(x_j)$ в рассматриваемом алгоритме Expectation Propagation (EP) совпадёт с формулой для пересчёта сообщения из фактора f_i в вершину x_j для алгоритма Loopy Belief Propagation [8] в аналогичном фактор-графе. Остальные формулы, используемые в LBP в точности совпадают с соответствующими формулами, полученными для EP.

	LBP	EP
$m_{f_i \to x_j}(x_j)$	$\int f(X_i) \prod_{x_k \in X_i^{\setminus j}} m_{x_k \to f_i}(x_k) dX_i^{\setminus j}$	$\frac{\text{proj} \left[\int f(X_i) \prod_{x_k \in X_i} m_{x_k \to f_i}(x_k) dX_i^{\setminus j} \right]}{m_{x_j \to f_i}(x_j)}$
$m_{x_j \to f_i}(x_j)$	$\prod_{k \neq i} m_{f_k \to x_j}(x_j)$	
$p(x_j)$	$\prod_{i: X_i \ni x_j} m_{f_i \to x_j}(x_j)$	

Таким образом, можно считать, что алгоритм EP является обобщением LBP на случай использования сложных распределений, для которых необходимо проецирование. Однако есть важная особенность: в случае, когда фактор-граф модели представляет собой дерево, можно организовать пересчёт сообщений в LBP таким образом, что за линейное от количества переменных в модели время можно точно вычислить все маргинальные распределения. Например, если фактор граф представляет собой цепочку как на рисунке 5, то в LBP достаточно один раз последовательно пересчитать все сообщения, ведущие слева направо, и ещё один раз пересчитать сообщения справа налево. Но при использовании сообщений из EP это свойство теряется, и какие-либо гарантии нахождения точных маргинальных распределений исчезают. На практике при применении EP к графам-цепочкам обычно несколько раз повторяют пересчёт сообщений, согласно расписанию для LBP, и, в результате, аппроксимация получается довольно точной.

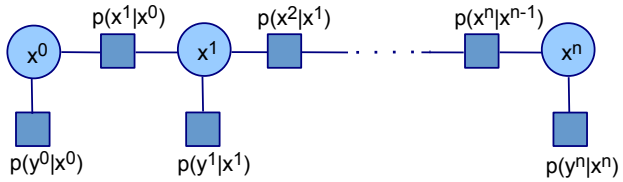


Рис. 5. Фактор-граф цепочка

3 Выбор аппроксимирующего семейства и метрики аппроксимации

В процессе построения алгоритма EP не конкретизировалось как именно стоит задавать базовое семейство $\mathcal{F}$ и каким образом следует осуществлять операцию проецирования на это семейство. В этом разделе будут описаны наиболее удобные и эффективные методы решения данного вопроса.

3.1 Экспоненциальное семейство распределений

Уже отмечалось, что для реализации пересчёта факторов в EP от базового семейства необходимо потребовать наличие возможности осуществлять операции умножения и деления, не выводящие за пределы семейства. Также, поскольку в задаче требуется упростить исходное распределение, стоит потребовать от семейства наличие эффективного способа вычисления нормировочной константы.

В расмотренной ранее задаче обнаружения сигнала в качестве такого семейства было выбрано пространство гауссиан. Ниже выписаны формулы, позволяющие перемножать и делить гауссианы.

$$\mathcal{N}(x; m_1, v_1)\mathcal{N}(x; m_2, v_2) = \mathcal{N}(m_1; m_2, v_1 + v_2)\mathcal{N}(x; m, v) \tag{18}$$
$$\text{где} \quad v = \left(\frac{1}{v_1} + \frac{1}{v_2}\right)^{-1}$$
$$m = v\left(\frac{m_1}{v_1} + \frac{m_2}{v_2}\right)$$

$$\mathcal{N}(x; m_1, v_1)/\mathcal{N}(x; m_2, v_2) = \frac{v_2\mathcal{N}(x; m, v)}{(v_2 - v_1)\mathcal{N}(m_1; m_2, v_2 - v_1)} \tag{19}$$
$$\text{где} \quad v = \left(\frac{1}{v_1} - \frac{1}{v_2}\right)^{-1}$$
$$m = v\left(\frac{m_1}{v_1} - \frac{m_2}{v_2}\right)$$

Как будет показано ниже, это не самый быстрый способ осуществления данных операций, но в такой форме он оказывается наиболее наглядным. Для начала отметим, что в формуле 19, если дисперсия делителя больше дисперсии частного, то у гауссианы в правой части в качестве дисперсии окажется отрицательное число. Формально это стоит воспринимать как гауссиану, хвосты которой увеличиваются до бесконечности при отдалении

от математического ожидания (Рис. 3а, слева). При этом о значении нормировочной константы можно не заботиться: после перемножения факторов для подсчёта искомых маргинальных распределений дисперсия снова будет положительной.

Удобно осуществлять умножение и деление можно также и над всем экспоненциальным семейством распределений (в которое в частности входят гауссианы). В стандартной форме распределения из данного семейства имеют следующий вид:

$$p(x|\theta) = \frac{1}{Z(\theta)} G(x) \exp\left(\theta^T g(x)\right)$$

$$\theta = [\theta_1; \ldots; \theta_n], \quad g(x) = [g_1(x); \ldots; g_n(x)]$$

Вектор θ принято называть ветором параметров распределения, а $g(x)$ — набором достаточных статистик. В нашем случае для использования элементов данного распределения в алгоритме EP от множителя $G(x)$ придётся отказаться, положив его тождественно равным 1. Иначе при перемножении распределений он будет возводиться в степень и потребуется вводить в семейство дополнительный параметр.

В таблице ниже представлены некоторые популярные распределения, которые также относятся к экспоненциальному семейству.

Распределение	Плотность	$g(x)$	θ
Бернулли	$q^x(1-q)^{1-x}$	x	$\ln \frac{q}{1-q}$
Нормальное	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$[x, x^2]$	$[\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}]$
Гамма	$\frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx)$	$[\ln(x), x]$	$[a-1, -b]$
Бета	$\frac{1}{\mathcal{B}(a,b)} x^{a-1}(1-x)^{b-1}$	$[\ln(x), \ln(1-x)]$	$[a-1, b-1]$
Пуассона	$\exp(-\lambda)\frac{\lambda^x}{x!}$	$[x, \ln\Gamma(x+1)]$	$[\ln(\lambda), -1]$
...	...	...	...

Очевидно, что для всех этих распределений умножение и деление плотностей вероятности сводится к сложению и вычитанию значений параметров распределения, а значение нормировочной константы является табличной величиной.

$$\frac{p(x|\theta_1) \cdot \ldots \cdot p(x|\theta_k)}{p(x|\theta_{k+1}) \cdot \ldots \cdot p(x|\theta_K)} = p\left(x|\theta = \sum_{i=1}^{k} \theta_i - \sum_{i=k+1}^{K} \theta_i\right) \tag{20}$$

3.2 Построение оператора проецирования

При описании решения задачи об обнаружении сигнала для проецирования сложного распределения на пространство гауссиан предлагалось воспользоваться методом приравнивания моментов распределений. Рассмотрим, какой смысл можно вложить в данную операцию и какими ещё способами можно осуществить проецирование:

1. $\text{proj}[p] = \arg\min_{q \in \mathcal{F}} KL(p \parallel q) = \arg\min_{q \in \mathcal{F}} \int p(x) \log \frac{p(x)}{q(x)} dx$

   Самый популярный метод проецирования и, как будет показано ниже, очень удобно осуществляемый при работе с экспоненциальным семейством.

2. $\text{proj}[p] = \arg\min_{q \in \mathcal{F}} D_\alpha(p \parallel q)$

   $D_\alpha(p \parallel q) = \frac{\int_x \alpha p(x) + (1-\alpha) q(x) - p(x)^\alpha q(x)^{1-\alpha} dx}{\alpha(1-\alpha)}$

   Минимизация α-дивергенции является обобщением предыдущего случая. Варьируя параметр α от минус до плюс бесконечности можно получать различные метрики, в том числе и KL-дивергенцию:

   (a) $D_{-1}(p \parallel q) = \frac{1}{2} \int_x \frac{(q(x)-p(x))^2}{p(x)} dx$

   (b) $\lim_{\alpha \to 0} D_\alpha(p \parallel q) = KL(q \parallel p)$

   (c) $D_{1/2}(p \parallel q) = 2 \int_x \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 dx$

   (d) $\lim_{\alpha \to 1} D_\alpha(p \parallel q) = KL(p \parallel q)$

(e) $D_2(p \parallel q) = \frac{1}{2}\int_x \frac{(p(x)-q(x))^2}{q(x)}dx$

В общем случае α-дивергенция обладает теми же свойствами, что и KL-дивергенция. Она неотрицательна, равна нулю только в случае одинаковых почти всюду аргументов, но при этом не является симметричной и может не удовлетворять неравенству треугольника.

3. $\text{proj}[p] = \arg\max_{q\in\mathcal{F}} \prod_{i=1}^{N_p} q(x_i); \quad \{x_i\}_{i=1}^{N_p} = \text{Sample}(p)$

   В некоторых случаях вместо минимизации некоторого функционала при проецировании бывает удобно выполнить следующий приём: из приближаемого распределения генерируется выборка, после чего из базового семейства выбирается распределение, максимизирующее правдоподобие этой выборки. Например, это приём может пригодиться при проецировании на смесь нескольких гауссиан. В этом случае максимизировать правдоподобие можно при помощи EM-алгоритма.

4. В зависимости от задачи можно использовать и другие произвольные, удобные для конкретной задачи метрики. Например, как уже отмечалось, приравнивать первый и второй момент у распределений...

Рассмотрим теперь, каким образом можно строить проекции на экспоненциальное семейство, при работе с перечисленными метриками. Для этого потребуется несколько утверждений.

Утверждение 1 *Пусть распределение $q(x)$ принадлежит экспоненциальному семейству: $q(x) = \frac{1}{Z(\theta)}\exp\left(\sum_{i=1}^{d}\theta_i g_i(x)\right)$. При этом функция $Z(\theta)$ непрерывна и дифференцируема по параметрам. Тогда $\mathbb{E}_q g_j(x) = \frac{\partial \log Z(\theta)}{\partial \theta_j}$*

Доказательство.

$$\frac{\partial \log Z(\theta)}{\partial \theta_j} = \frac{1}{Z(\theta)}\frac{\partial}{\partial \theta_j}Z(\theta) = \frac{1}{Z(\theta)}\frac{\partial}{\partial \theta_j}\int_x \exp\left(\sum_{i=1}^{d}\theta_i g_i(x)\right)dx =$$

$$= \frac{1}{Z(\theta)}\int_x g_j(x)\exp\left(\sum_{i=1}^{d}\theta_i g_i(x)\right)dx = \int_x g_j(x)q(x)dx = \mathbb{E}_q g_j(x)$$

■

Утверждение 2 *Пусть $q(x)$ принадлежит экспоненциальному семейству. Тогда $q = \arg\min_q KL(p \parallel q) \Leftrightarrow \mathbb{E}_p g_j(x) = \mathbb{E}_q g_j(x) \quad \forall j = 1\dots d$*

Доказательство.

$$KL(p \parallel q) = \int_x p(x)\log\frac{p(x)}{q(x)}dx = const - \int_x p(x)\log q(x)dx =$$

$$= const + \int_x p(x)\log Z(\theta)dx - \int_x p(x)\sum_{i=1}^{d}\theta_i g_i(x)dx =$$

$$= const + \log Z(\theta) - \sum_{i=1}^{d}\theta_i\int_x p(x)g_i(x)dx$$

Необходимым условием минимизации функционала является равенство нулю всех его частных производных. Учитывая утверждение 1, получаем:
$\frac{\partial}{\partial \theta_j}KL(p \parallel q) = \mathbb{E}_q g_j(x) - \mathbb{E}_p g_j(x) = 0 \quad \forall j = 1\dots d$ ■

Таким образом минимизация обратной KL-дивергенции свелась к приравниванию математических ожиданий достаточных статистик $g_i(x)$. В частности у нормального распределения достаточными статистиками являются x и x^2, так что используемый до этого метод приравнивания моментов эквивалентен минимизации обратной KL-дивергенции.

Утверждение 3 *Пусть распределение $q_\theta(x)$ непрерывно зависит от параметра θ.*
Тогда, если $\alpha \neq 0$:
$\hat{\theta}$ особая точка $D_\alpha(p(x) \parallel q_{\hat{\theta}}(x))$ $\Leftrightarrow$
$\hat{\theta}$ особая точка $KL\left(p(x)^\alpha q_{\hat{\theta}}(x)^{1-\alpha} \parallel q_{\hat{\theta}}(x)\right)$

Под особыми точками здесь понимаются точки нулевого градиента функии Лагранжа данной задачи минимизации. Такими точками являются локальные экстремумы и сёдла минимизируемого функционала, однако для выбранных функционалов особые точки как правило оказываются локальными минимумами.

Доказательство данного утверждения приведено в статье [5]. В нашем случае данный факт позволяет строить проекции не только для KL-дивергенции, но и для других метрик из α-семейства. Минимизировать $D_\alpha(p \parallel q)$ можно повторяя итерационный процесс:

1. $q'(x) = \arg\min_{q'} KL\left(p(x)^\alpha q(x)^{1-\alpha} \parallel q'(x)\right)$
2. $q(x)^{new} = q(x)^\varepsilon q'(x)^{1-\varepsilon}$

Заметим, что данный метод не подходит для минимизации прямой KL-дивергенции ($\alpha = 0$). На практике, для минимизации этого функционала можно минимизировать метрику $D_\alpha(p \parallel q)$ для малого α.

Видно, что существует довольно много способов организации проецирования на экспоненциальное семейство. Возникает вопрос: для чего нужна минимизация различных метрик, если всегда с лёгкостью можно использовать обратную KL-дивергенцию $KL(p \parallel q)$? Причин несколько:

1. При работе с моделью из нескольких переменных для аппроксимации фактора осуществлялась локализация контекста и проводилась минимизация локальной дивергенции между распределениями. В статье [5] продемонстрировано, что для глобальной минимизации α-дивергенции с некоторым заданным параметром при локальной минимизации иногда стоит выбирать другое значение α;

2. В статье [6] описаны проблемы, возникающие при минимизации обратной KL-дивергенции для фактора $f(a, b, c) = \mathbb{I}[a \times b = c]$. При пересчёте сообщений в переменную-множитель проецируемое распределение оказывается бимодальным и, например, гауссиана минимизирующая $KL(p \parallel q)$ будет покрывать обе моды, что при дальнейшей работе алгоритма приведёт к неограниченному росту дисперсии у переменных-множителей и неадекватному результату. С другой стороны минимизация $KL(q \parallel p)$ позволяет избежать данного эффекта и при проецировании приближать только одну из мод;

3. Достаточные статистики у элементов экспоненциального семейства могут иметь достаточно сложный вид и вычисление их математического ожидания у аппроксимируемого распределения для минимизации $KL(p \parallel q)$ может оказаться вычислительно сложной задачей. В этом случае при проецировании может оказаться разумным приравнивание других статистик. Например, при проецировании на Бета-распределение вместо вычисления математических ожиданий $\mathbb{E}_p \log(x)$ и $\mathbb{E}_p \log(1 - x)$, которые может быть тяжело рассчитать у аппроксимируемого распределения, можно вычислить первый и второй момент распределения $p(x)$.

4 Библиотека Infer.NET

Несложно заметить, что при использовании алгоритма Expectation Propagation в вероятностной модели, заданной фактор-графом, основная проблема состоит в построении проекций сложных функций на выбранное семейство распределений. Этот прооцесс приходится осуществлять только при вычислении сообщений $m_{f_i \to x_j}(x_j)$ из фактора в переменную. Остальные сообщения и маргинальные распределения скрытых переменных как правило вычисляются без проблем. При этом заметим, что процесс построения каждой проекций зависит только от одного приближаемого фактора и семейства, на которое осуществляется проецирование, и не зависит от положения этого фактора в фактор-графе.

На практике множество факторов, используемых для построения вероятностных моделей, оказывается не слишком большим. В большинстве случаев оно содержит набор различных ограничений на значения переменных в модели и описание функциональных зависимостей между переменными. Зависимости как правило можно представить в виде суперпозиции базовых арифметических операций и элементарных функций, таких как экспонента, сигмоида или логарифм.

Подходящее базовое семейство тоже обычно можно удачно выбрать из довольно небольшого множества распределений. В большинстве моделей можно ограничится использованием наиболее популярных представителей

экспоненциального семейства распределений: нормальное распределение, бета, гамма, дискретное, распределение Пуассона и Дирихле.

Таким образом множество типичных ситуаций, требующих в алгоритме EP построения проекции, оказывается конечным. Это даёт возможность, вывести все необходимые формулы пересчёта сообщений в данных ситуациях заранее, и в дальнейшем не задумываться о процессе пересчёта сообщений при проектировании модели.

В настоящее время существует несколько программных библиотек, позволяющих строить вероятностные модели, используя заданный набор допустимых факторов и базовых распределений. Одной из них является, используемая мною при написании данной работы библиотека *Microsoft Infer.Net* [3].

Главная особенность данной библиотеки заключается в использовании случайных величин вместо привычных константных переменных. Например, пользователь может создать две переменных: x_n, имеющую нормальное распределение с некоторыми параметрами, и x_g, имеющую гамма-распределение. После чего использовать данные переменные для описания новой переменной y, не задумываясь о том каким будет распределение y. Ниже представлен фрагмент кода на языке C#, иллюстрирующего данную возможность.

```
Variable<double> x_n = Variable.GaussianFromMeanAndVariance(6, 1);
Variable<double> x_g = Variable.GammaFromShapeAndScale(1, 1);
Variable<double> y = Variable.GaussianFromMeanAndPrecision(0, x_g) + x_n;
Variable.ConstrainTrue(y > 0.5);
```

В результате в построенной вероятностной модели y будет иметь распределение

$$p(y) = \mathbb{I}(y > 0.5) \cdot \left(\mathcal{N}(y|0, \frac{1}{x_g}) + x_n \right)$$

При этом пользователю не требуется выбирать аппроксимацию для этой переменной. Система сделает это автоматически.

В настоящий момент библиотека поддерживает 3 режима передачи сообщений: минимизирующие прямую KL-дивергенцию (вариационный вывод), обратную KL-дивергенцию (классический вариант EP) и построение аппроксимации на основе сэмплирования из приближаемого распределения. Не каждый из этих режимов поддерживает использование всех реализованых в библиотеке факторов, однако по мере выпуска более новых версий список возможностей, предоставляемых пользователю, стабильно увеличивается.

Как и любая другая библиотека, выпущенная для платформы *.NET Framework*, Infer.NET позволяет использовать свои возможности в программах на любом языке программирования, поддерживающем общеязыковую среду исполнения *Common Language Runtime (CLR)*. Например, такими языками являются C#, F#, C++/CLI, VB .NET и JScript .NET, IronPython, Perl и PHP.

5 Задача прогнозирования результативности игроков

В этом разделе будет продемонстрировано, как полученный алгоритм распространения ожидания можно применять для решения практической задачи прогнозирования результативности участников турнира. Необходимость в решении такой задачи может возникнуть, например, в следующих ситуациях:

- При построении рейтинга участников турнира для определения победителей соревнований, или для отбора наиболее опытных игроков в состав сборной;
- Во многих online-играх участник соревнуется с другими, выбранными автоматически, соперниками. Для повышения интереса игроков соперников стоит выбирать максимально похожими по силе.

Данная задача прогнозирования сводится к построению содержательной математической модели, описывающей навыки игроков, и вычислению требуемых прогнозов в рамках данной модели. Ниже представлены основные требования к такой модели для составления рейтинга игроков в Halo в системе Xbox Live [2]:

1. Мастерство игроков изменяется со временем;
2. Игроки участвуют как в личных, так и командных соревнованиях;
3. В одной игре может соревноваться произвольное число команд;
4. Игра допускает возможность возникновения ничьих между некоторыми командами;
5. Требуется обрабатывать большие и непрерывно растущие объёмы данных.

5.1 Модель Эло

Первая модель определения навыков была составлена в 1939 году и использовалась для вычисления рейтингов участников шахматных турниров. У модели было много недостатков, которые часто приводили к неадекватным результатам. В 1959 году американским профессором физики Арпадом Эло была разработана новая модель, учитывающая вероятностную оценку исхода партий, и оказавшаяся более эффективной при обработке результатов шахматных турниров.

Согласно модели Эло каждому игроку соответствует неизвестный уровень навыков, skill ($s_i \in \mathbb{R}$). Во время участия в турнире игроку i соответствует уровень его результативности, являющийся случайной величиной p_i, зависящей от s_i и заданного параметра модели β.

$$p_i \sim \mathcal{N}(p_i; s_i, \beta^2) \tag{21}$$

Результат партии представляется в виде дискретной переменной y, принимавшей значения $\{1, 0, -1\}$ в случае победы, ничьей и поражения первого игрока соответственно. Считалось, что игрок 1 выиграл у игрока 2, если $p_1 > p_2$. Согласно построенной модели вероятность того, что первый игрок выиграет у второго равна $\Phi\left(\frac{s_1 - s_2}{\sqrt{2}\beta}\right)$. По результатам партии производится обновление значения ожидаемого уровня навыков:

$$\begin{aligned} s_1 &\leftarrow s_1 + y\Delta \\ s_2 &\leftarrow s_2 - y\Delta \\ \text{где} \quad \Delta &= \alpha\beta\sqrt{\pi}\left(\frac{y+1}{2} - \Phi\left(\frac{s_1 - s_2}{\sqrt{2}\beta}\right)\right) \end{aligned}$$

Таким образом, если, например, первый игрок имеет высокий рейтинг, а второй — новичок с низким рейтингом, то в результате победы первого рейтинги практически не изменятся, но в случае победы второго игрока значение поправки Δ будет достаточно большим.

Разумеется у данной модели остаётся много недостатков. Например, в случае ничьей обновления рейтингов вообще не происходит, даже если играли новичок и гроссмейстер.

5.2 Модель TrueSkill

В этом разделе будет описана разработанная в 2007 году модель TrueSkill, устраняющая большую часть недостатков модели Эло, а также позволяющая использовать для составления рейтингов результаты турниров в которых участвовало сразу несколько игроков.

5.2.1 Описание модели

В модели TrueSkill для описания навыков игрока также используется вещественное число, однако теперь оно не задаётся явным образом, а представляется как случайная величина s_i, принимающая значения на всей вещественной прямой и заданная своим математическим ожиданием и дисперсией. Данную случайную величину удобно аппроксимировать гауссианой.

Добавление в модель дисперсии позволяет оценивать достоверность нашей оценки рейтинга игрока: у новых игроков, сыгравших мало партий, дисперсия устанавливается большой. Но со временем о результативности игрока накапливается больше статистической информации, и дисперсию величины s_i можно считать незначительной. Помимо учёта количества сыграных партий модель учитывает тот факт, что опыт игрока мог меняться со временем, даже если он не участвовал в турнирах. Для формализации этого явления в модели предполагается, что дисперсия величины s_i со временем увеличивается. Таким образом, если считать, что время принимает дискретные значения, априорные представления о прогнозе рейтинга игрока i можно описать следующим образом:

$$s_i^0 \sim \mathcal{N}(s_i^0; \mu_0, \gamma_0^2) \tag{22}$$

$$s_i^{t+1} \sim \mathcal{N}(s_i^{t+1}; s_i^t, \gamma^2) \tag{23}$$

где μ_0, γ_0 и γ — параметры модели, а s_i^t — значение величины s_i в момент времени t. Описание модели одной партии также усложнилось. Как и в модели Эло предполагается, что результативность p_i игрока является случайной величиной с математическим ожиданием равным s_i (см. формулу 21). Однако теперь в одной партии

могут соревноваться сразу несколько игроков, разбившихся на команды и результативность команды оценивается как суммарная результативность всех её участников в данной партии:

$$t_k = \sum_{i \in T_k} p_i \tag{24}$$

Особый интерес представляет вопрос: как в рамках полученной вероятностной модели описать наблюдаемый исход партии в случае, если одновременно соревновалось несколько команд? В большинстве существующих в настоящее время спортивных и компьютерных игр исход партии или матча можно представить в виде перестановки участвовавших команд (для личных соревнований можно считать, что каждая команда состоит из одного человека). При этом, если команда a заняла более высокую позицию, чем команда b, то считается, что $t_a > t_b$. Для описания возможности возникновения ничьих данное неравенство следует сделать более строгим:

- Если $t_a > t_b + \varepsilon$, то команда a победила команду b;
- Если $t_a + \varepsilon < t_b$, то команда a проиграла команде b;
- Если $|t_a - t_b| \leqslant \varepsilon$, то команды a и b заняли одинаковую позицию в проведённой партии.

Пользуясь транзитивностью оператора сравнения, выполнения неравенств записанных выше можно потребовать не от всех пар команд, а только от занявших соседние позиции в перестановке с результатами партии. При этом, конечно, не совсем корректным будет описание ситуаций при которых несколько команд занимают одинаковое место, но как правило при работе с реальными данными этим можно пренебречь.

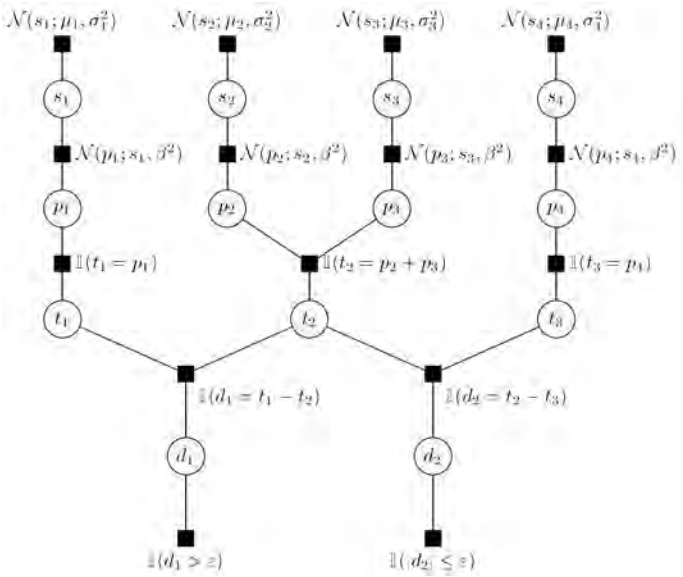


Рис. 6. Фактор-граф для одной партии в модели TrueSkill

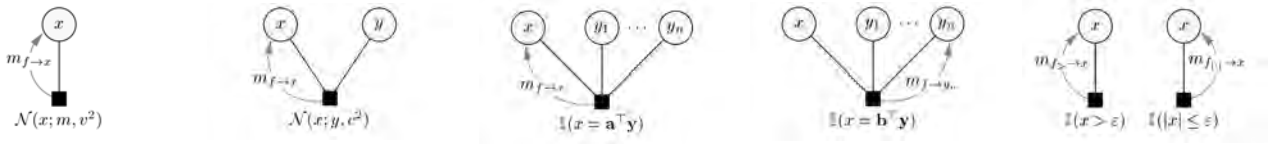
Рассмотрим как будет выглядеть фактор-граф описанной модели на примере одной партии (Рис. 6). В данном примере в партии участвуют четыре игрока, поделившиеся на три команды (в первой команде игрок под номером 1, во второй — с номерами 2 и 3, а в третьей один игрок с номером 4). По результатам партии команда с номером 1 заняла первое место, а второе место поделили команды 2 и 3. Таким образом, согласно принятым соглашениям, верно следующее:

$$t_1 - t_2 > \varepsilon$$
$$|t_2 - t_3| \leqslant \varepsilon$$

Считается, что перед началом партии было известно априорное распределение навыков игроков. Для обновления рейтингов согласно заданной модели необходимо найти апостериорное распределение переменных s_i при условии известных результатов партии.

Для оценки апостериорного распределения воспользуемся алгоритмом Expectation Propagation для построенного фактор графа. Для этого необходимо научиться пересчитывать сообщения для всех введённых в модель факторов.

5.2.2 Вывод формул для пересчёта сообщений



(a) Фактор 1 (b) Фактор 2 (c) Фактор 3(1) (d) Фактор 3(2) (e) Фактор 4

Рис. 7. Передача сообщений в модели TrueSkill

Как уже отмечалось, в данной модели будет удобно в качестве базового семейства выбрать пространство гауссиан. Формулы для вычисления сообщений из вершины в фактор и для оценки маргинальных распределений имеют стандартный для EP вид, и не предсталяют интереса. Главная трудность при работе с любой моделью заключается в получении формул для пересчёта сообщений из факторов в переменные.

Всего в построенной модели встречается 4 принципиально различающихся типа факторов. Первый, самый простой, соответствует введению априорного распределения на переменные s_i(Рис. 7a). Для вычисления сообщения из данного фактора в переменную воспользуемся формулой 17.

$$\mathbf{m_{f\to x}}(\mathbf{x}) = \frac{\text{proj}\left[\mathcal{N}(x; m, v^2) m_{x\to f}\right]}{m_{x\to f}} = \mathcal{N}(\mathbf{x}; \mathbf{m}, \mathbf{v^2}) \tag{25}$$

Пользуясь тем, что под оператором проецирования стоит произведение гауссиан (которое тоже является гауссианой), от проецирования можно избавиться. В итоге сообщение принимает тривиальный вид.

На втором слое фактор-графа модели расположены факторы-гауссианы, зависящие от двух переменных (Рис 7b). Снова воспользуемся формулой для пересчёта сообщений в EP:

$$\mathbf{m_{f\to x}}(\mathbf{x}) = \frac{\text{proj}\left[m_{x\to f}\int \mathcal{N}(x; y, c^2) m_{y\to f} dy\right]}{m_{x\to f}} = \frac{\text{proj}\left[m_{x\to f}\int \mathcal{N}(y; x, c^2)\mathcal{N}(y; m_y, v_y^2) dy\right]}{m_{x\to f}} =$$

$$= \frac{\text{proj}\left[m_{x\to f}\mathcal{N}(x; m_y, c^2 + v_y^2)\int \mathcal{N}(y; \hat{m}, \hat{v}^2) dy\right]}{m_{x\to f}} = \mathcal{N}(\mathbf{x}; \mathbf{m_y}, \mathbf{v_y^2} + \mathbf{c^2}) \tag{26}$$

Здесь потребовалось воспользоваться формулой 18 для перемножения гауссиан под интегралом. После чего оказалось, что под оператором проецирования снова стоит гауссиана, и снова сообщение из фактора в переменную можно вычислить точно. Заметим, что поскольку выражения $\mathcal{N}(x; y, v^2)$ и $\mathcal{N}(y; x, v^2)$ являются эквивалентными, сообщение из данного фактора во вторую переменную вычисляется аналогично.

На третьем и четвёртом слое построенного фактор-графа располагаются факторы, которые можно описать в виде, показанном на (Рис. 7c). В данной модели координаты вектора $\mathbf{a}$ принимают значения 1 и -1, но формулу для пересчёта сообщений несложно вывести и для произвольных значений координат.

$$\mathbf{m_{f\to x}}(\mathbf{x}) = \frac{\text{proj}\left[m_{x\to f}\int_{\mathbb{I}[x=\mathbf{a}^T\mathbf{y}]}\prod_{i=1}^{n} m_{y_i\to f} d\mathbf{y}\right]}{m_{x\to f}} = \frac{\text{proj}\left[m_{x\to f}\mathcal{N}\left(x; \sum_{i=1}^{n} a_i m_{y_i}, \sum_{i=1}^{n} a_i^2 v_{y_i}^2\right)\right]}{m_{x\to f}} =$$

$$= \mathcal{N}\left(\mathbf{x}; \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{a_i m_{y_i}}, \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{a_i^2 v_{y_i}^2}\right) \tag{27}$$

Интеграл, возникающий в данном выражении, соответствует некоторому распределению на x равному линейной комбинации независимых гауссиан. По свойству нормального распределения, линейная комбинация независимых гауссиан также является гауссианой с указанными в формуле параметрами. В результате под опрератором проецирования снова оказывается произведение двух гауссиан и сообщение из данного фактора снова можно вычислить точно.

Сообщения из данного фактора в переменные y_i можно вычислить аналогично (Рис. 7d). Для этого заметим, что

$$\mathbb{I}\left[x = \mathbf{b}^T\mathbf{y}\right] = \mathbb{I}\left[y_i = \mathbf{a}^T[y_1, \ldots, y_{i-1}, x, y_{i+1}, \ldots, y_n]\right]$$

$$\text{где} \quad \mathbf{a}^T = \frac{1}{b_i}[-b_1, \ldots, -b_{i-1}, +1, -b_{i+1}, \ldots, b_n]$$

Следовательно, задачу можно свести к предыдущей.

Наиболее трудными для вычисления оказываются сообщения из факторов с нижнего уровня построенной графической модели (Рис. 7e). По формуле 17 соответствующие сообщения должны вычисляться следующим образом:

$$\mathbf{m}_{\mathbf{f}\to\mathbf{x}}(\mathbf{x}) = \frac{\text{proj}\left[\mathcal{N}(x; m, \sigma^2)\mathbb{I}_i(x)\right]}{\mathcal{N}(x; m, \sigma^2)}$$

$$\text{где} \quad \mathbb{I}_1(x) = \mathbb{I}(x > \varepsilon) \qquad \mathbb{I}_2(x) = \mathbb{I}(|x| \leqslant \varepsilon)$$

Под оператором проецирования оказалась усечённая гауссиана. Для построения проекции на пространство гауссиан будем минимизировать обратную KL-дивергенцию.

$$\text{proj}\left[\mathcal{N}(x; m, \sigma^2)\mathbb{I}_i(x)\right] = \arg\min_{q\in\mathcal{F}} KL(\mathcal{N}(x; m, \sigma^2)\mathbb{I}_i(x) \parallel q(x)) = \mathcal{N}(x; \hat{\mu}_i, {\hat{\sigma}_i}^2) \tag{28}$$

По доказанному утверждению 2 для осуществления минимизации необходимо приравнять математические ожидания достаточных статистик выбранного базового семейства. В данной ситуации нужно потребовать выполнения следующих равенств

$$\mathbb{E}_{q_i} x = \hat{\mu}_i = \mathbb{E}_{p_i} x \tag{29}$$

$$\mathbb{E}_{q_i} x^2 = {\hat{\mu}_i}^2 + {\hat{\sigma}_i}^2 = \mathbb{E}_{p_i} x^2 \tag{30}$$

Вычислим необходимые математические ожидания статистик для индикатора $\mathbb{I}_1$:

$$M_1^0(\varepsilon) \equiv Z_{p_1} = \int_\varepsilon^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = \{t = \frac{x-\mu}{\sigma}\} = \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = 1 - \Phi\left(\frac{\varepsilon-\mu}{\sigma}\right) \tag{31}$$

$$\begin{aligned} M_1^1(\varepsilon) \equiv Z_{p_1}\mathbb{E}_{p_1} x = \int_\varepsilon^{+\infty} x \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = \{t = \frac{x-\mu}{\sigma}\} = \\ = \sigma \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} d\frac{t^2}{2} + \mu \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = \\ = \sigma^2 \mathcal{N}(\varepsilon; \mu, \sigma^2) + \mu\left(1 - \Phi\left(\frac{\varepsilon-\mu}{\sigma}\right)\right) \end{aligned} \tag{32}$$

$$\begin{aligned} M_1^2(\varepsilon) \equiv Z_{p_1}\mathbb{E}_{p_1} x^2 = \int_\varepsilon^{+\infty} x^2 \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = \{t = \frac{x-\mu}{\sigma}\} = \\ = \frac{\sigma^2}{\sqrt{2\pi}} \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} t^2 e^{-\frac{t^2}{2}} dt + 2\sigma\mu \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} \frac{t}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt + \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} \frac{\mu^2}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = \\ = \frac{\sigma^2}{\sqrt{2\pi}} \int_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty} \left[t^2 e^{-\frac{t^2}{2}} - e^{-\frac{t^2}{2}} + e^{-\frac{t^2}{2}}\right] dt + 2\sigma^2\mu\mathcal{N}(\varepsilon; \mu, \sigma^2) + \mu^2 M_1^0 = \\ = \sigma^2 \left[-\frac{t}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\Big|_{\frac{\varepsilon-\mu}{\sigma}}^{+\infty}\right] + 2\sigma^2\mu\mathcal{N}(\varepsilon; \mu, \sigma^2) + (\mu^2 + \sigma^2) M_1^0 = \\ = \sigma^2(\varepsilon + \mu)\mathcal{N}(\varepsilon; \mu, \sigma^2) + (\mu^2 + \sigma^2)\left(1 - \Phi\left(\frac{\varepsilon-\mu}{\sigma}\right)\right) \end{aligned} \tag{33}$$

Пользуясь равенствами 29 — 33, параметры проекции $\hat{\mu}_1$ и $\hat{\sigma}_1$ можно найти из уравнений:

$$\hat{\mu}_1 = \frac{M_1^1(\varepsilon)}{M_1^0(\varepsilon)} = \frac{\sigma^2\mathcal{N}(\varepsilon; \mu, \sigma^2)}{\Phi\left(\frac{\mu-\varepsilon}{\sigma}\right)} + \mu \tag{34}$$

$$\hat{\mu}_1^2 + \hat{\sigma}_1^2 = \frac{M_1^2(\varepsilon)}{M_1^0(\varepsilon)} = \frac{\sigma^2(\varepsilon + \mu)\mathcal{N}(\varepsilon; \mu, \sigma^2)}{\Phi\left(\frac{\mu-\varepsilon}{\sigma}\right)} + (\mu^2 + \sigma^2) \tag{35}$$

Математические ожидания статистик для индикатора $\mathbb{I}_2$ вычисляются аналогично. Достаточно заметить, что

$$M_2^0(\varepsilon) = M_1^0(-\varepsilon) - M_1^0(\varepsilon)$$
$$M_2^1(\varepsilon) = M_1^1(-\varepsilon) - M_1^1(\varepsilon)$$
$$M_2^2(\varepsilon) = M_1^2(-\varepsilon) - M_1^2(\varepsilon)$$

Наконец, научившись строить проекцию усечённой гауссианы, легко получить искомое значение сообщения $\mathbf{m}_{\mathbf{f}_i \to \mathbf{x}}$ как частное двух гауссиан $\mathcal{N}(x; \hat{\mu}_i, \hat{\sigma}_i{}^2)$ и $\mathcal{N}(x; m, \sigma^2)$ по формуле 19.

5.2.3 Составление расписания передачи сообщений

Фактор-граф на (Рис. 6), составленный для данной модельной партии, представляет собой дерево. Если бы сообщения из введёных факторов можно было вычислять без использования оператора проецирования, то все необходимые маргинальные распределения можно было бы получить за линейное от размера графа время при помощи алгоритма LBP. Однако, как было только что установлено, точному вычислению поддаются все сообщения, кроме тех, которые исходят из факторов с нижнего уровня, поэтому LBP применить не получится.

Следуя логике EP, сообщения в графе необходимо итерационно пересчитывать до сходимости, но в данном случае можно избежать полного пересчёта. Заметим, что сообщения, ведущие вниз между первыми тремя слоями факторов не зависят от сообщений на нижних уровнях. Аналогично, сообщения, ведущие между тремя первыми слоями факторов вверх не влияют ни на какие другие сообщения. Таким образом, их можно вычислять только 1 раз как в LBP, а итерационному пересчёту подлежат только сообщения проходящие внутри минимальноо связного подграфа, содержащего все приближенно вычисляющиеся сообщения [2].

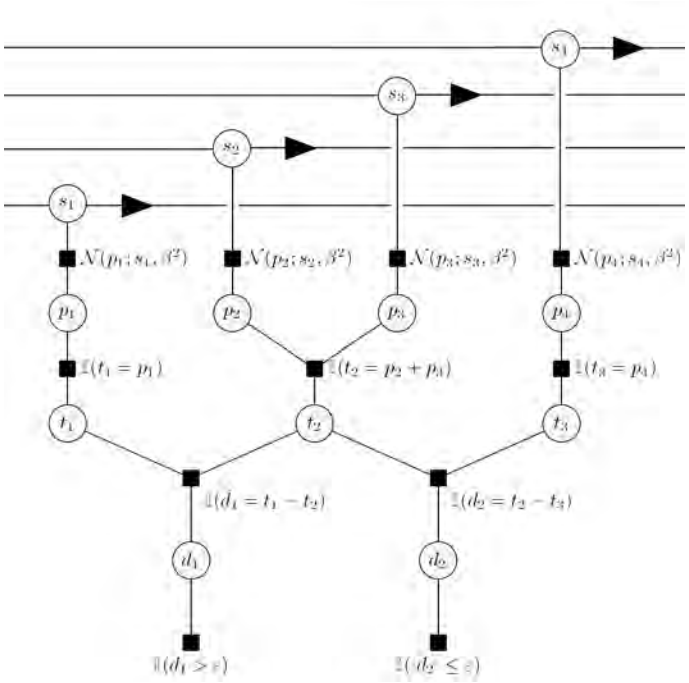


Рис. 8. Часть фактор-графа в модели TrueSkill, соответствующая одной партии

Для работы с базовой моделью это наблюдение помогает слабо: до сих пор рассматривался фактор-граф только для одной партии, однако согласно описанию модели, фактор-графы всех партий связаны между собой через переменные s_i^t (Рис. 8). Таким образом, чтобы честно оценить распределения на искомые переменные, отвечающие за навыки игроков, необходимо итерационно пересчитывать все сообщения в огромном графе, связывающем все партии. С учётом реальных объемов данных и того факта, что с течением времени будет возникать необходимость этот граф достраивать, учитывая результаты новых партий, такой подход неприменим.

Наиболее простое решение данной проблемы, используемое при построении рейтингов игроков в Halo, состоит в следующем: вместо того чтобы честно объединять фактор-графы всех матчей в один, каждый матч рассматривается независимо (так, как это было сделано в процессе вывода формул для сообщений). При этом в качестве

априорного распределения на переменные s_i используется постериорное распределение на соответствующие переменные, полученное по результатам прошлого матча. При этом если между матчами прошло определённое количество времени, то у априорного распределения на соответствующую величину увеличивается дисперсия согласно формуле 23.

5.3 Модель TrueSkill Through Time

У описанного выше подхода к проблеме пересчёта сообщений есть существенный недостаток: он не использует новую информацию для корректировки старых прогнозов. Рассмотрим следующую ситуацию: пускай в некоторый момент времени состоялась партия, в которой сразились всего два игрока a и b, про которых до этого ничего не было известно. Пусть игрок a победил, в этом случае среднее значение s_a станет чуть выше начального μ_0, а среднее значение s_b станет чуть меньше. Пусть потом, через некоторое небольшое время, игрок b выиграет у игрока c, являющего профессионалом и имеющего очень большой рейтинг. В результате, естественно, среднее значение s_b значительно увеличится, но при этом с прогнозом рейтинга игрока a ничего не произойдёт, хотя известно, что он только что одолел заведомо неслабого соперника — игрока b.

В модификации к базовому алгоритму TrueSkill Through Time (TTT) для решения отмеченного недостатка предлагается очень красивый приём. Вместо того, чтобы для каждого игрока хранить отдельную переменную s_i^t, предлагается объединить все переменные, соответствующие одному моменту времени, в одну многомерную переменную $S^t = \{s_i^t\}_{i=1}^{|Players|}$, а часть фактор-графа, соответствующую одной партии представить в виде одного сложного фактора (Рис. 9).

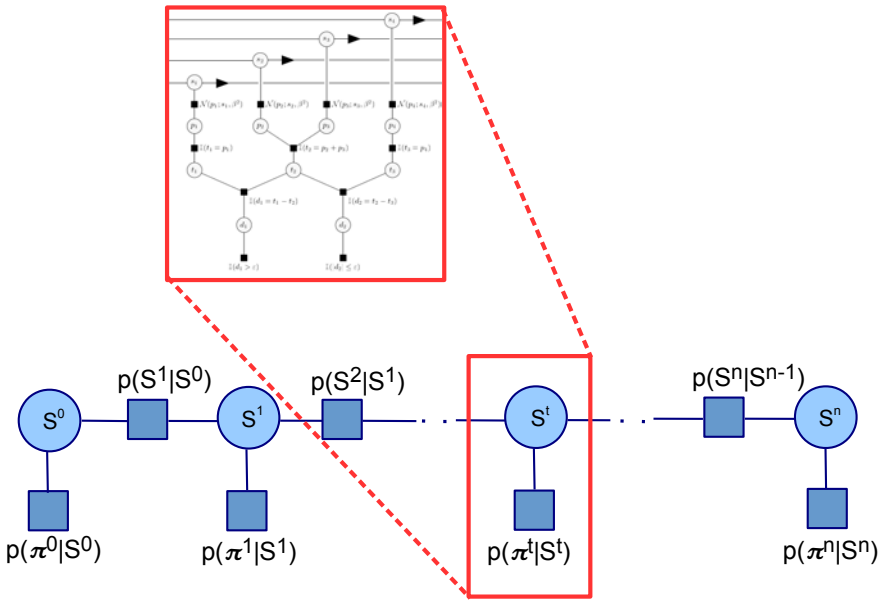


Рис. 9. TrueSkill Through Time

В результате фактор граф, содержащий скрытые переменные S^t, будет иметь вид марковской цепи. Как уже отмечалось раннее, экспериментально установлено, что для таких графов существует расписание сообщений, позволяющие быстро добиваться сходимости алгортма EP и достаточно точно оценивать маргинальные распределения (для этого нужно несколько раз повторить пересчёт сообщений по оптимальной схеме для алгоритма LBP).

5.4 Модель TTT-D

В статье [7] для демонстрации качества работы алгоритма TTT авторы используют результаты крупных шахматных турниров с 1850 по 2006 год. На графике (Рис. 10) показаны изменения среднего значения и дисперсии прогнозируемого рейтинга для наиболее известных шахматистов за указанный период.

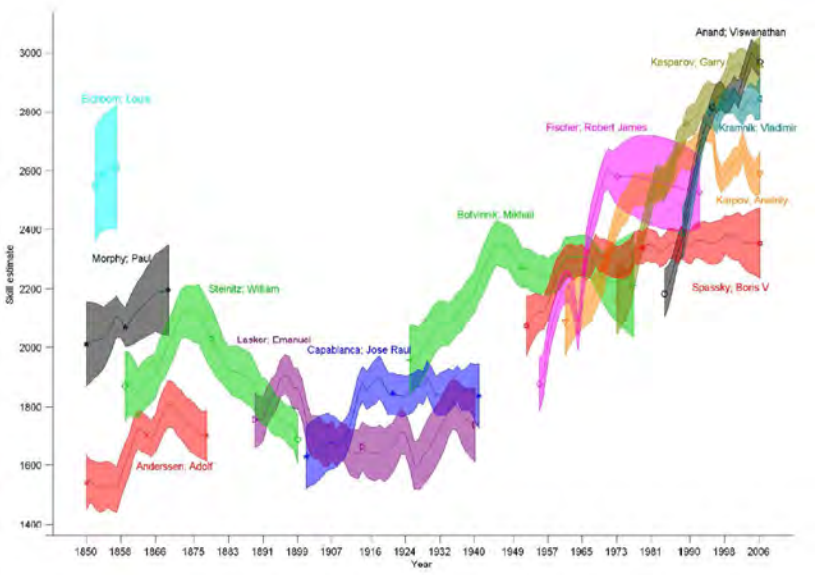


Рис. 10. TrueSkill Through Time для ранжирования шахматистов

Полученные при помощи модели результаты действительно хорошо согласуются с реальностью, но при этом сама модель TrueSkill достаточно универсальна и не позволяет учесть особенностей описания навыков в конкретной игре. Например, в шахматах помимо основного навыка игры оказывается важным умение сводить вничью партию при игре с более сильным соперником. Эти особенности учтены в усложнённой модели «TrueSkill Through Time with Individual Draw Margins» (TTT-D), в которой каждого игрока игрока помимо случайной величины s_i^t характеризует его индивидуальный навык ε_i^t сводить игру в ничью, являющийся неотрицательной случайной величиной.

$$\varepsilon_i^0 \sim \mathcal{N}(\varepsilon_i^0; \nu_0, \varsigma_0^2)\mathbb{I}(\varepsilon_i^0 > 0)$$
$$\varepsilon_i^{t+1} \sim \mathcal{N}(\varepsilon_i^{t+1}; \varepsilon_i^t, \varsigma^2)\mathbb{I}(\varepsilon_i^{t+1} > 0)$$

Теперь для того чтобы игрок a победил игрока b, требуется чтобы выполнилось неравенство:

$$p_a > p_b + \varepsilon_b \tag{36}$$

На (Рис. 11) показано как будет выглядеть фактор-граф для одной партии с участием двух шахматистов при различных исходах состязания. Отметим, что несмотря на кажущееся усложнение, в модель не добавилось ни одного нового вида факторов. Как уже упоминалось при описании библиотеки Infer.NET, такая ситуация является типичной при построении вероятностных моделей.

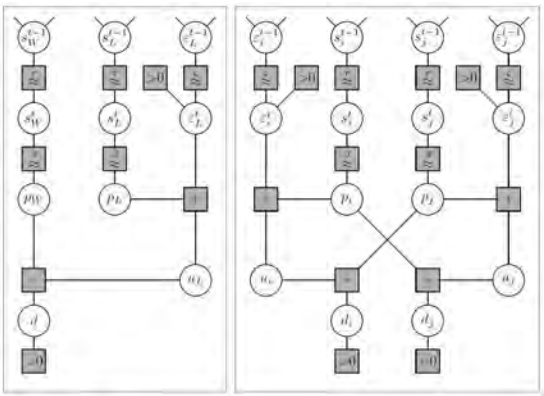


Рис. 11. Фактор-граф для одной партии модели TTT-D. Игрок W победил игрока L (слева). Игрок I сыграл вничью с игроком J (справа).

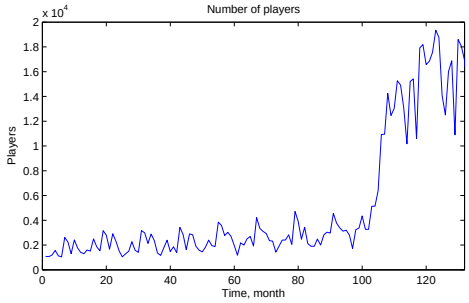
6 Эксперименты с моделью TTT-D

За время написания выпускной квалификационной работы был освоен процесс описания вероятностных моделей на языке C# с помощью библиотеки Infer.NET версии 2.6 и проведён ряд экспериментов по тестированию модели прогнозирования результативности игроков «TrueSkill Through Time with Individual Draw Margins».

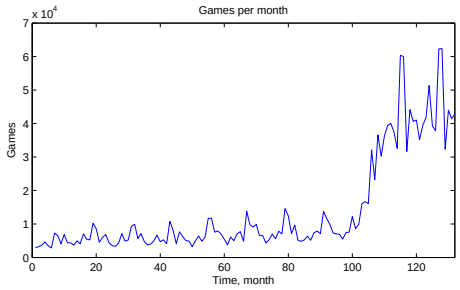
В качестве исходных данных для тестирования модели использовалась информация о результатах шахматных турниров за временной период протяжённостью 11 лет, предоставляемая международной шахматной федерацией (FIDE). Данные взяты с сайта kaggle.com, на котором проводилось соревнование по прогнозированию результатов шахматных турниров [1].

Предоставляемая участникам соревнования база данных содержит описание приблизительно 1.8 миллиона партий в которых приняли участие более 54000 человек. Для каждой партии известны идентификационные номера участвовавших шахматистов, цвет фигур каждого из игроков, дата проведения партии с точностью до месяца и результат партии (победа белых/ победа чёрных/ ничья).

Для обучения модели использовалась информация о результатах шахматных партий за временной промежуток протяжённостью 10 лет (первые 120 месяцев в предоставленной базе данных). Для тестирования использовалась информация о результатах партий в последующие 12 месяцев.



(a) Зависимость количества участников соревнований от номера месяца



(b) Зависимость количества партий от номера месяца

Результатом обучения модели является совокупность апостериорных распределений на переменные s_i^t и ε_i^t. При предсказании победителя партии предполагается, что игрок a победил игрока b, если оказалось выполнено неравенство 36, в котором в качестве значений p_a, p_b и ε_b взяты математические ожидания соответствующих случайных величин.

На рис. 12a и 12b показано распределение партий и количества участников по времени (для каждого месяца посчитано количество партий и количество игроков, сыгравших хотя бы одну партию в этом месяце). Видно, что после 110-го месяца данные статистики существенно возрастают. Таким образом, получается, что для большинства игроков, участвовавших в партиях, отнесённых к тестовой выборке, информация об их предыдущих результатах практически отсутствует.

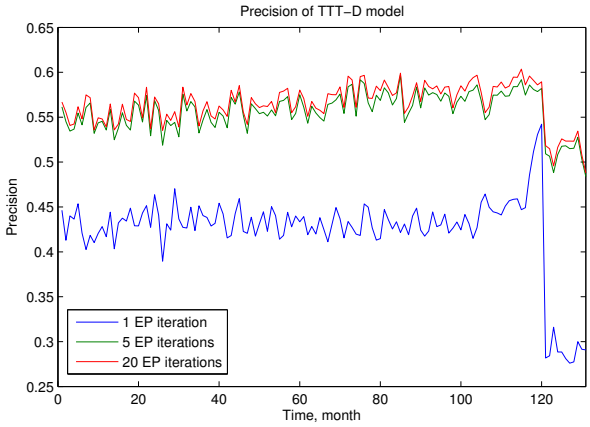


Рис. 13. Доля угаданных результатов партий в каждом месяце

На рис. 13 представлена зависимость доли угадывания моделью результатов партии от номера месяца. Сравнение проведено для моделей, обученных с помощью 1, 5 и 20 итераций алгоритма Expectation Propagation. На графиках видно, что для партий, попавших в обучающую и тестовую выборки, вероятность угадывания правильного ответа существенно отличатеся. Однако, уже 5 итераций алгоритма оказывается достаточно для построения модели, способной угадывать результат будущих партий как минимум в половине случаев (для алгоритма, выдающего случайный ответ, вероятность угадывания равна $1/3$).

Стоит отметить, что качество моделей, обученных с помощью 5 и 20 итераций EP оказалось практически одинаковым. При этом данный алгоритм оказался довольно быстрым: время работы одной итерации метода на полном фактор-графе не превышает двух минут при распараллеливании вычислений на 6 потоков на процессоре частотой 2.9 Ghz. Основную трудность представляет хранение фактор-графа в оперативной памяти: на полной выборке размер, построенного с помощью Infer.NET графа, составил 8.8 GiB.

7 Прогнозирование результатов футбольных матчей

7.1 Недостатки моделей серии TrueSkill

Как показывают эксперименты, модели серии TrueSkill достаточно хорошо соответствуют реальности и позволяют строить относительно точные прогнозы для результатов спортивных состязаний и компьютерных игр [2]. Однако они содержат ряд недостатков:

- **Плохая практическая интерпретация навыков:** оценки навыков, получающиеся при работе алгоритма могут принимать отрицательные значения. Такие оценки имеют плохую практическую интерпретацию, так как команда не может играть себе во вред;
- **Неограниченный рост дисперсии:** дисперсия игрока, сыгравшего несколько матчей и после этого оставившего соревнования на длительное время может оказаться существенно больше дисперсии оценки на навык игрока, которой не сыграл ни одной игры. Также модели серии TrueSkill не позволяют угадывать неизвестные результаты матчей, проходившие до получения первой информации об учавствовавших в них командах;
- **Использование слишком тривиальной информации о результатах матчей:** в качестве результа партии существующие модели учитывают только победителя. При этом во многих играх можно получить гораздо больше полезной информации о матче (например, счёт партии или информацию о турнире к которому этот матч относился);
- **Тривиальный результат предсказаний:** Модели серии TrueSkill позволяют предсказывать только победителя партии, но не могут дать информацию об ожидаемом детализованном результате (например, счёт партии или распределение полученных командой очков в партии между её участниками).

От большей части указанных выше недостатков мне удалось избавиться в разработанной модификации для модели TTT-D под названием FootballSkill.

7.2 Модель FootballSkill

Реализованная в рамках данной работы вероятностная модель FootballSkill в первую очередь направлена на прогнозирование результатов футбольных матчей. Данная модель позволяет учитывать счёт матча, а также более интерпретабельно описывает мастерство команд. Ниже представлено детальное описание разработанной модели:

Мастерство игрока:

В данной модели в роли участников соревнований выступают неделимые команды (в частности, такое описание подходит и для личных соревнований в которых в команда состоит всего из одного участника). Навыки каждой команды описываются двумя величинами: сила атаки (a_i) и сила обороны (d_i). Как и в моделях серии TrueSkill данные показатели могут меняться со временем по нормальному закону. Более того, для каждого момента времени и для каждого из навыков команды учитывается априорное распределение на данный навык, задаваемое произведением двух факторов: $f_{prior_1}(x) = \mathcal{N}(x|\mu_0, \tau_0^2)$ и $f_{prior_1}(x) = \mathbb{I}(x > 0)$. Итого:

$$a_i^t \sim \mathcal{N}(a_i^t|\mu_0, \tau_0^2)\mathbb{I}(s_i^t > 0) \qquad d_i^t \sim \mathcal{N}(d_i^t|\mu_0, \tau_0^2)\mathbb{I}(d_i^t > 0) \tag{37}$$

$$a_i^{t+1} \sim \mathcal{N}(a_i^{t+1}|a_i^t, \tau^2) \qquad d_i^{t+1} \sim \mathcal{N}(d_i^{t+1}|d_i^t, \tau^2) \tag{38}$$

Результативность игрока:

Как и в моделях серии TrueSkill в разработанной модели предполагается, что в каждом матче эффективность каждой из характеристик игрока представляет собой случайную величину, распределённую по нормальному закону, с математическим ожиданием, равным истинному значению соответсвующей характеристики:

$$\hat{a}_i \sim \mathcal{N}(\hat{a}_i | a_i, \beta_a^2) \qquad \hat{d}_i \sim \mathcal{N}(\hat{d}_i | d_i, \beta_d^2) \tag{39}$$

Ниже будет видно, что с математической точки зрения, значение имеет только сумма дисперсий $\beta_a^2 + \beta_d^2$. Так как ожидается, что обе характеристики команд (сила атаки и сила защиты) имеют примерно одинаковую значимость, то и при настройке параметров модели предполагалось, что $\beta_a^2 = \beta_d^2 = \beta^2$.

Счёт игрока:

Количество очков набранных командой в матче (например, количество забитых командой голов в футболе) рассчитывается как разница эффективного значения силы атаки данной команд и эффективной силы защиты противника в данном матче.

$$g_x = \hat{a}_x - \hat{d}_y \qquad g_y = \hat{a}_y - \hat{d}_x \tag{40}$$

Видно, что по сути величина g_x имеет нормальное распределение $\mathcal{N}(g_x | a_x - d_y, \beta_a^2 + \beta_d^2)$ и действительно зависит только от суммы шумовых дисперсий.

Результаты матча:

На этапе обучения модели нам известен счёт каждого матча и мы можем просто зафиксировать наблюдаемое значение переменных g_x и g_y

$$Observed(g_x) = score_x \qquad Observed(g_y) = score_y \tag{41}$$

На этапе тестирования возможны два варианта:

1. Если имеется необходимость предсказать конкретный счёт в матче, то вероятность того, что команда x наберёт k очков можно оценить как

   $$p_x^k = P(k - 0.5 < g_x \leqslant k + 0.5) \tag{42}$$

   Поскольку случайная переменная g_x в результате применения алгоритма распространения ожидания будет иметь нормальное распределение, требуемая вероятность p_x^k легко вычисляется для любого целого k;

2. Если достаточно предсказать победителя партии, то для этого можно рассчитать вероятность того, что одна из команд набрала хотя бы на 1 очко больше чем другая:

   $$p_{x\,win} = P(g_x > g_y + 1) \qquad p_{y\,win} = P(g_y > g_x + 1) \tag{43}$$

   Соответственно, вероятность ничьей равна:

   $$p_{draw} = 1 - p_{x\,win} - p_{y\,win} \tag{44}$$

Предсказываемым результатом матча является наиболее вероятный исход в модели.

Заметим, что от добавления в фактор-граф, построенный при обучении модели, подграфов, соответствующих матчам из тестовой выборки, маргинальное распределение у переменных, описывающих навыки команд, не изменится (в добавленных подграфах нигде не используется информация о результатах тестовых матчей. Поэтому сообщения, исходящие из данных подграфов, будут представлять собой вырожденные равномерные распределения). Таким образом мы можем одновременно осуществлять пересчёт сообщений в подграфах для обучения и тестиирования.

8 Эксперименты с моделью FootballSkill

8.1 Анализ исходных данных

Для проведения экспериментов с разработанной моделью FootballSkill была составлена выбрка из матчей наиболее крупных чемпионатов по футболу России и СССР за период с 1936 по 2014 год. Информация о данных матчах была взята с сайта [9]. В подготовленную выборку с информацией о матчах включены все матчи высшего

и первого дивизиона, кубка России и Советского Союза, а также несколько тысяч товарищеских матчей. После выкачивания соответствующих протоколов с указанного сайта пришлось провести дополнительную обработку имеющихся данных, чтобы исключить все аннулированные матчи, а также указать все факты переименования футбольных клубов в период с 1936 по 2014 год. Всего полученная выборка содержит информацию о 92 122 матчах в которых приняли участие 1 347 различных команд.

О каждом матче имеется следующая информация:

- Названия участвовавших команд;
- Счёт матча по результатам двух периодов;
- Дата проведения матча;
- Номер команды, являвшейся хозяином стадиона.

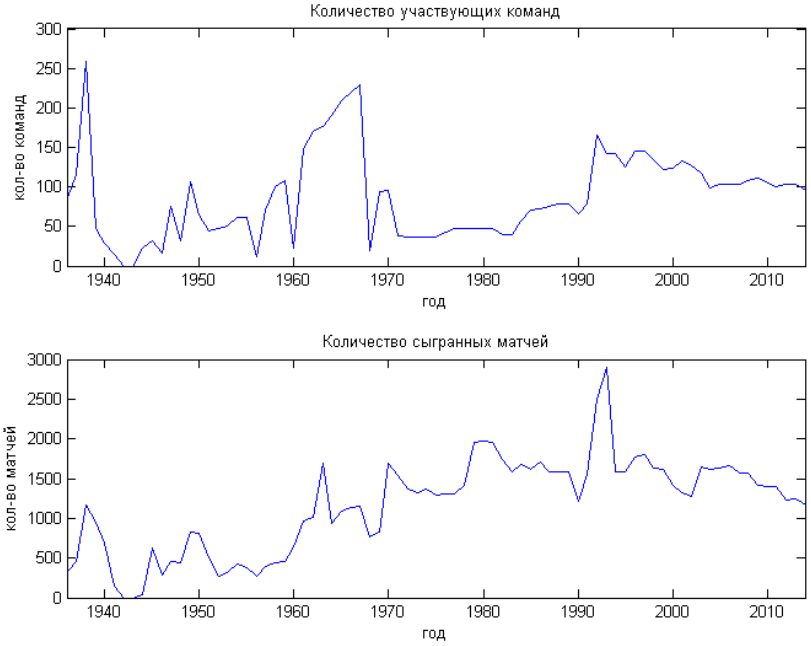


Рис. 14. Анализ имеющихся данных о футбольных матчах

На рис. 14 представлены графики с основными описательными характеристиками имеющихся данных. Видно, что в 1942 и 1943 годах матчей не проводилось из-за участия СССР во Второй Мировой войне. Также заметно сильное увеличение количества участвующих команд в 1992 году после распада Советского Союза и реформы системы футбольных чемпионатов.

8.2 Настройка гиперпараметров модели

Разработанная модель содержит следующие гиперпараметры:

- μ_0 – математическое ожидание в априорном распределении навыков;
- τ_0 – стандартное отклонение в априорном распределении навыков;
- τ – стандартное отклонение навыков во времени;
- β – стандартное отклонение эффективных значений навыков в матче.

Результат каждого матча зависит только от разности между навыками атаки и обороны соперничающих команд. Поэтому от прибавления константы ко всем значениям навыков всех команд прогнозы матчей не должны

меняться. Из-за проектирования на пространство гауссиан факторы, отвечающие за описание априорного распределения, по сути задают одну гауссиану с некоторым неотрицательным математическим ожиданием. Таким образом, зафиксировав $\mu_0 = 0$ и настраивая параметр τ_0, мы можем задать в качестве априорного распределение с любой дисперсией и некоторым положительным математическим ожиданием. Точное значение τ_0 практически не влияет на результат обучения модели, важен только порядок данной величины.

Для каждого фиксированного τ_0 был проведён подбор оптимального значения β. При настройке данного параметра было составлено 2 выборки, каждая из которых содержала случайную половину матчей за период с 2005 по 2013 год. В предположениях, что за указанный период навыки команд менялись не слишком сильно ($\tau = 0$), при помощи двухвыборочной кросс-валидации, максимизирующей усреднённое правдоподобие на тестовых выборках, была получена пара оптимальных значений τ_0 и β ($\tau_0 = 100$; $\beta = 2$).

При полученных значениях τ_0 и β был проведён подбор оптимального значения τ. Для этого на две выборки случайным образом были разбиты матчи за период с 1936 по 2013 год. В качестве единицы измерения времени был выбран промежуток, равный 1 году. Как и в предыдущем пункте, с помощью двухвыборочной кросс-валидации было найдено значение τ, максимизирующее усреднённое правдоподобие результатов матчей в тестовой выборке. В качестве оптимального было выбрано значение $\tau = 0.5$.

Возможные значения параметров τ_0, τ и β на этапе кросс-валидации перебирались по следующей сетке:

- $\tau_0 \in \{1; 10; 100; 1000\}$;
- $\tau \in \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1; 1.5; 2.0; 2.5; 3.0; 4.5; 5.0; 6; 7; 8; 9; 10; 15; 20; 25; 30\}$;
- $\beta \in \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1; 1.5; 2.0; 2.5; 3.0; 4.5; 5.0; 6; 7; 8; 9; 10; 15; 20; 25; 30\}$;

Стоит отметить, что во всех экспериментах значения параметров, соответствующие максимуму правдоподобия на тестовой выборке практически не отличались от значений параметров, соответсвующих максимальной доле верно угаданных результатов матчей.

8.3 Оценка навыков команд

На рис. 15 показаны найденные при помощи разработанной модели оценки навыков для 6 наиболее известных команд Российской футбольной Премьер-Лиги. Видно, что наибольшее значение навыка атаки демонстрируют команды «Спартак (Москва)» и «Зенит (Санкт-Петербург.)». Команда «ЦСКА» стабильно демонстрирует высокий уровень защиты, но довольно низкий уровнь атаки. Данные наблюдения хорошо согласуются с существующими общепринятыми представлениями.

Рис. 15. Изменение навыков команд со временем

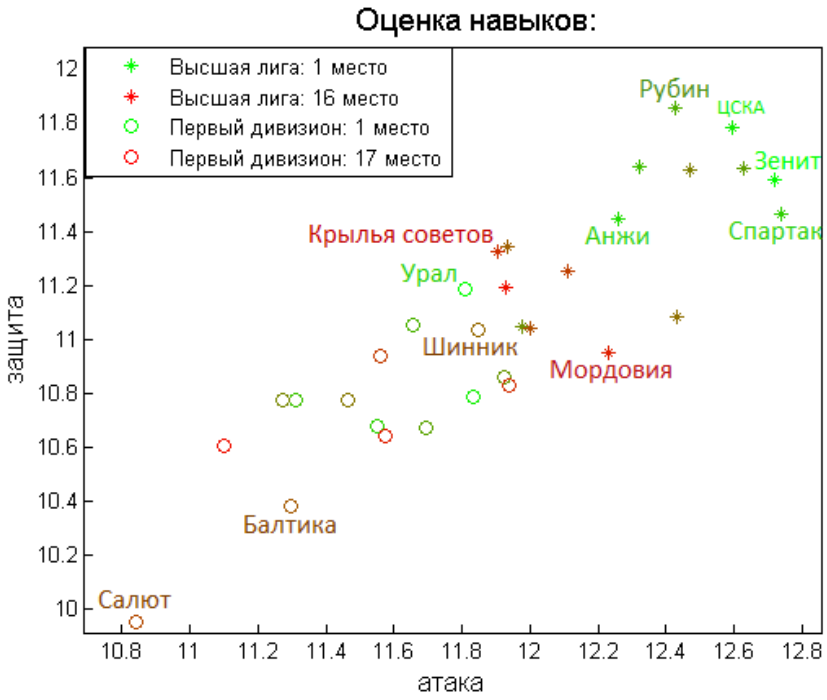


Рис. 16. Объединение рейтингов высшего и первого дивизионов: команды, выступавшие в высшей лиге, отмечены на графике звёздочками, а команды из первого дивизиона — кружочками. Цветом показано положение команд в рейтинге соответствующего дивизиона (от зелёного цвета, соответствующего первому месту в лиге, до красного— соответствующего последнему месту).

Важной особенностью рейтингов, полученных при помощи обучения модели FootballSkill является то, что они позволяют сравнить в единой шкале команды, которые никогда прежде не играли в одном чемпионате. Например, на рис. 16 показано как соотносятся рейтинги команд высшего и первого дивизиона по состоянию на 2012-2013 год. Видно, что команды высшей лиги в среднем обгоняют первый дивизион как по навыкам атаки, так и по навыкам защиты, однако аутсайдеры высшей лиги уже могут уступать по некоторым характеристикам лидирующим командам более слабого дивизиона.

8.4 Модель FS-HomeAdvantage

На рис. 17 представлено распределение среднего количества забитых и пропущенных мячей по всем имеющимся командам. Видно, что в большинстве случаев команда, играющая дома, в среднем забивает больше мячей, чем при игре в гостях. И аналогично при игре на домашнем стадионе практически все команды в среднем пропускают меньше голов.

В таблице ниже указаны усреднённые по командам средние значения забитых и пропущенных мячей:

	забито	пропущено
дома	1.316	1.228
в гостях	0.814	1.918

Этот эффект легко можно учесть при построении модели матча путём прибавления некоторой константы h к результативности атаки и защиты команды, играющей на домашнем стадионе. Оптимальное значение h ($h = 0.5$) было подобрано при помощи двухвыборочной кросс-валидации на матчах за период с 1936 по 2013 год.

Введение в модель данной константы также позволило снизить средний ожидаемый разброс эффективных значений атаки и обороны в матчах. По результатам повторной кросс-валидации для модели, учитывающей влияние игры на домашнем стадионе, в качестве оптимального значения β было выбрано значение 1.5.

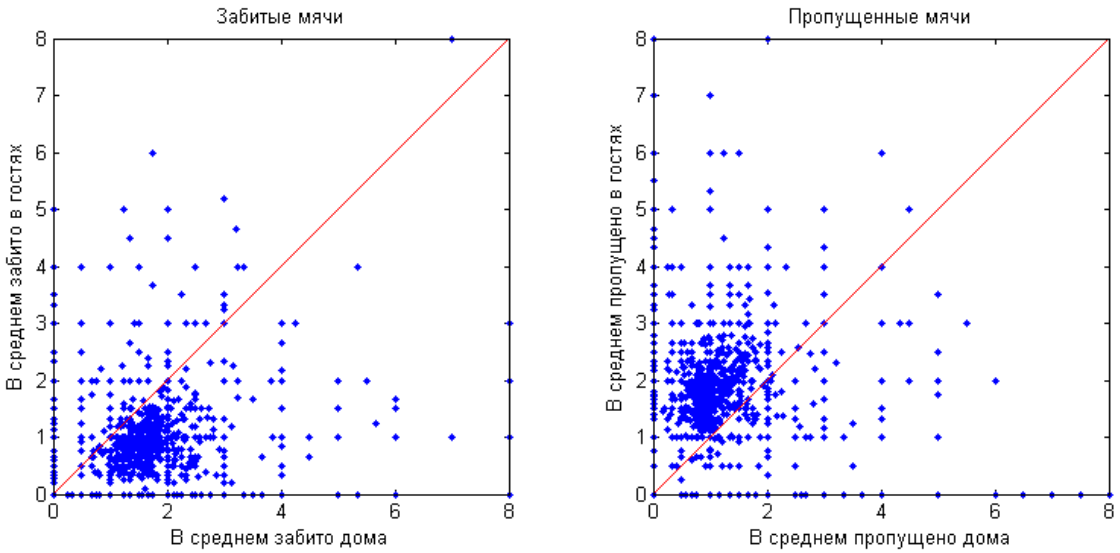


Рис. 17. Средние значения забитых/пропущенных мячей на домашнем стадионе и в гостях

8.5 Точность прогнозирования

При проведении экспериментов было проведено сравнение точности предсказания результатов разработанных моделей FootballSkill и FS-HomeAdvantage с моделями TTT-D и ELO на имеющихся матчах чемпионатов России и Советсткого Союза.

Сравнение проводилось для двух режимов:

- **Интерполяция результатов:** обучение проводилось на случайной половине матчей за период с 1936 по 2013 год. Оставшиеся матчи использовались в качестве тестовой выборки;
- **Экстраполяция результатов:** обучение моделей проводилось на всех матчах с 1936 по 2013 год. Для тестирования были использованы матчи 2014 года.

Достигнутая точность предсказания для указанных режимов тестирования представлена в таблице:

	1936-2013	2014
ELO	45.07%	47.27%
TTT-D	46.32%	45.39%
FootballSkill	49.02%	51.53%
FS-HomeAdvantage	56.62%	51.76%

Видно, что разарботанные модели FootballSkill и FS-HomeAdvantage показали лучшее качество по сравнению с аналогичной моделью TTT-D, не использующей специфической информации о матчах, и классической моделью ELO. Таким образом, использование дополнительной информации об исходе матчей позволило существенно повысить точность прогнозирования.

Полученные результаты довольно сильно похожи на аналогичные результаты для предсказания шахматных турниров. Одна из главных причин, по которой точность прогнозирования оказывается на столько низкой, заключается в том, что по структуре большинства турниров часто возникает ситуация при которой в один год сначала команда A выигрывает у команды B, после чего команда A команде B проигрывает. Так как с точки зрения модели никаких различий между данными матчами не было, при предсказании результата хотя бы одного из этих матчей модель FootballSkill гарантированно ошибается.

9 Заключение

За время написания выпускной квалификационной работы мною был подробно изучен алгоритм распространения ожидания, являющийся одним из базовых алгоритмов вывода в вероятностных моделях в библиотеке Infer.NET и освоены возможности библиотеки Infer.NET версии 2.6.

По результатам выполнения ВКР мне удалось разработать и обучить на реальных данных вероятностную модель для предсказания результатов футбольных матчей и оценки способностей футбольных команд. Для вывода искомых распределений был разработан алгоритм передачи сообщений на основе изученного метода распространения ожидания.

За счёт использования дополнительной информации о структуре матча мне удалось разработать вероятностную модель, которая показывают более высокое качество прогнозирования результатов спортивных соревнований по сравнению с моделями серии TrueSkill и классической моделью ELO.

Список литературы

1. Deloitte/FIDE Chess Rating Challenge. — 2011. — URL: www.kaggle.com/c/ChessRatings2.
2. *Herbrich R.*, *Minka T.*, *Graepel T.* Trueskill™: A Bayesian skill rating system // Advances in Neural Information Processing Systems. — 2006. — С. 569—576.
3. Infer.NET 2.6 / T. Minka [и др.]. — 2014. — Microsoft Research Cambridge. http://research.microsoft.com/infernet.
4. *Minka T. P.* Expectation propagation for approximate Bayesian inference // Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence. — Morgan Kaufmann Publishers Inc. 2001. — С. 362—369.
5. *Minka T.* [и др.] Divergence measures and message passing: тех. отч. / Technical report, Microsoft Research. — 2005.
6. *Stern D. H.*, *Herbrich R.*, *Graepel T.* Matchbox: large scale online bayesian recommendations // Proceedings of the 18th international conference on World wide web. — ACM. 2009. — С. 111—120.
7. Trueskill through time: Revisiting the history of chess / P. Dangauthier [и др.] // Advances in Neural Information Processing Systems. — 2007. — С. 337—344.
8. *Yedidia J. S.*, *Freeman W. T.*, *Weiss Y.* Understanding belief propagation and its generalizations // Exploring artificial intelligence in the new millennium. — 2003. — Т. 8. — С. 236—239.
9. Протоколы футбольных матчей. — 2015. — URL: http://fc-dynamo.ru/champ/.