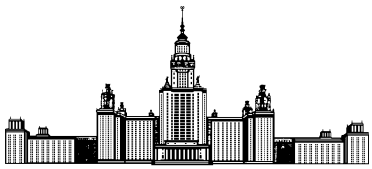
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Комбинаторные оценки обобщающей способности и их применение для построения композиций линейных классификаторов»

Выполнил:

студент 5 курса 517 группы

Соколов Евгений Андреевич

Научный руководитель:

д.ф-м.н., доцент

Воронцов Константин Вячеславович

Москва, 2013

Содержание

1 Введение

При решении задач распознавания образов, восстановления регрессии, прогнозирования всегда возникает проблема выбора по неполной информации. Имея лишь конечную обучающую выборку объектов, требуется из заданного множества алгоритмов выбрать алгоритм, который ошибался бы как можно реже не только на объектах наблюдаемой обучающей выборки, но и на объектах скрытой контрольной выборки, которая в момент выбора алгоритма ещё неизвестна. Если частота ошибок на контрольной выборке оказывается значительно выше, чем на обучающей, то говорят, что произошло «переобучение» (overtraining) или «переподгонка» (overfitting) алгоритма — он слишком хорошо описывает конкретные данные, но не обладает способностью к обобщению этих данных, не восстанавливает порождающую их зависимость и не пригоден для построения прогнозов.

Проблема обобщающей способности является ключевой и в то же время наиболее сложной в машинном обучении. Ее изучению посвящена *теория статистического обучения*. Одной из основных ее задач является получение *оценок вероятности переобучения*, предсказывающих качество работы алгоритма классификации на скрытой контрольной выборке. В своем большинстве такие оценки выводятся с помощью неравенств концентрации меры [19]. Среди них имеются оценки, учитывающие размерность семейства алгоритмов (VC-оценки [23]), свойства метода обучения (например, локальные свойства метода обучения [14]), и различные характеристики обучающей выборки X (например, нормализованные отступы [15, 16]). Однако, даже самые современные оценки являются слишком завышенными, и пока не существует ни одного примера их успешного практического применения применения.

Комбинаторная теория переобучения — это новое направление в теории статистического обучения, развиваемое К.В. Воронцовым и его учениками. В его рамках впервые удалось получить точные оценки вероятности переобучения для ряда модельных семейств алгоритмов классификации: слоёв и интервалов булева куба, монотонных и унимодальных цепей [24] и многомерных сетей [8], хэммингових шаров и некоторых их разреженных подмножеств [4]. Теоретико-групповой подход [10] позволяет получать точные оценки для семейств с произвольными симметриями.

Однако центральным направлением развития комбинаторного подхода является его применение для анализа переобучения реальных семейств, а также улучшение с его помощью существующих методов обучения алгоритмов классификации.

В [1] было экспериментально показано, что вероятность переобучения стандартных методов классификации (нейронных сетей, решающих деревьев, ближайшего соседа) на реальных задачах может быть аппроксимирована с помощью монотонных сетей подходящей размерности. В [2] предложен способ «подмены» реального семейства, возникающего при выборе порогов в узлах решающих деревьев, монотонными или унимодальными сетями, и показано, что использование точных оценок вероятности переобучения в критерии ветвления повышает обобщающую способность решающих деревьев.

В [25] была описана структура семейства логических конъюнкций, что позволило вычислять для него комбинаторные оценки вероятности переобучения. В той же работе комбинаторный подход был применен для уменьшения переобучения конъюнктивных закономерностей в логических алгоритмах, что стало первым примером его использования для улучшения методов классификации.

Данная работа посвящена применению комбинаторного подхода к семейству линейных классификаторов.

В первом разделе вводятся формальные определения из комбинаторной теории, включая вероятность переобучения, метод обучения и граф расслоения-связности.

Во втором разделе доказывается новая оценка вероятности переобучения, учитывающая связи между несравнимыми алгоритмами, что позволяет повысить точность по сравнению с существующими комбинаторными оценками.

В третьем разделе описывается структура семейства линейных классификаторов, а также предлагается алгоритм для обхода графа расслоения-связности данного семейства. Данный алгоритм делает возможным вычисление комбинаторных оценок для линейных классификаторов.

В четвертом разделе обсуждается задача приближенного вычисления комбинаторных оценок, и предлагается использование случайных блужданий для решения данной задачи. Подходы, предложенные в данном разделе, позволяют быстро вы-

числять комбинаторные оценки, что значительно упрощает их практическое применение.

Пятый раздел посвящен применению комбинаторных оценок для построения композиций линейных классификаторов. Приводятся результаты экспериментов на реальных задачах, а также производится сравнение с PAC-Bayes оценками, которые являются одними из лучших на данный момент в теории статистического обучения.

1.1 Основные определения

Пусть задано конечное множество $\mathbb{X} = \{x_1, \ldots, x_L\}$, называемое *генеральной выборкой*. Пусть также задано множество $\mathbb{A}$, элементы которого называют *алгоритмами*. Также предполагается, что задана бинарная функция $I : \mathbb{A} \times \mathbb{X} \to \{0, 1\}$, называемая *индикатором ошибки*. Функция $I(a, x)$ принимает значение 1, если алгоритм a допускает ошибку на объекте x, и значение 0 в противном случае.

Вектором ошибок алгоритма a называется бинарный вектор $\vec{a} = (I(a, x_i))_{i=1}^{L}$. Везде далее мы будем отождествлять алгоритм с его вектором ошибок. Также будем считать, что $\mathbb{A}$ — это множество бинарных векторов.

Метод обучения — это функция, строящая по подмножеству полной выборки алгоритм из заданного семейства: $\mu : 2^{\mathbb{X}} \to \mathbb{A}$.

Будем рассматривать задачу обучения по прецедентам в следующей постановке. Пусть на этапе обучения известна *обучающая выборка* $X \subset \mathbb{X}$ длины ℓ. По обучающей выборке с помощью заданного метода обучения μ выбирается алгоритм μX из семейства $\mathbb{A}$. После того, как обучение закончено, становится известной *скрытая выборка* $\bar{X} = \mathbb{X} \setminus X$, и к ней применяется выбранный алгоритм μX. Длину контрольной выборки будем обозначать через $k = L - \ell$.

Числом ошибок алгоритма a на выборке $X \subset \mathbb{X}$ называют величину

$$n(a, X) = \sum_{x \in X} I(a, x).$$

Частотой ошибок алгоритма a на выборке $X \subset \mathbb{X}$ называется величина

$$\nu(a, X) = \frac{n(a, X)}{|X|}.$$

Уклонением частот ошибок алгоритма a на двух выборках X и $\bar{X} = \mathbb{X} \setminus X$ называется величина

$$\delta(a, X) = \nu(a, \bar{X}) - \nu(a, X).$$

Пусть задан некоторый вещественный параметр $\varepsilon \in [0, 1)$, называемый *порогом переобучения*. Говорят, что алгоритм a переобучается на разбиении $(X, \bar{X})$, если

$$\delta(a, X) \geqslant \varepsilon.$$

Аналогично, метод μ переобучается на разбиении $(X, \bar{X})$, если

$$\delta(\mu X, X) \geqslant \varepsilon.$$

Наше единственное вероятностное предположение будет заключаться в том, что все разбиения генеральной выборки на обучающую и контрольную равновероятны. Исходя из этого предположения, определим *вероятность переобучения* — центральное понятие в данной работе.

Определение 1. *Вероятностью переобучения метода μ называется доля разбиений генеральной выборки на обучающую и контрольную, на которых метод обучения переобучается:*

$$Q_\varepsilon(\mu, \mathbb{X}) \equiv \mathsf{P}[\delta(\mu X, X) \geqslant \varepsilon] = \frac{1}{C_L^\ell} \sum_{(X, \bar{X})} [\delta(\mu X, X) \geqslant \varepsilon].$$

Обращением верхней оценки на вероятность переобучения $Q_\varepsilon \leqslant \eta(\varepsilon)$ называется неравенство $\nu(\mu X, \bar{X}) - \nu(\mu X, X) \leqslant \varepsilon(\eta)$, которое выполняется с вероятностью не менее $1-\eta$, где $\varepsilon(\eta)$ — функция, обратная к $\eta(\varepsilon)$. *Медианой верхней оценки* $Q_\varepsilon \leqslant \eta(\varepsilon)$ называется ее обращение при $\eta = 1/2$.

Определим некоторые комбинаторные величины, которые понадобятся нам в дальнейшем.

Гипергеометрическая функция вероятности:

$$h_L^{\ell,m}(s) = \frac{C_m^s C_{L-m}^{\ell-s}}{C_L^\ell}.$$

Гипергеометрическая функция распределения:

$$H_L^{\ell,m}(s) = \sum_{i=0}^{\min(s,\ell,m)} h_L^{\ell,m}(i).$$

Подробное описание этих величин можно найти в [?].

В данной работе большое внимание будет уделено семейству *линейных классификаторов*. Пусть объекты выборки представляют собой точки в евклидовом пространстве: $\mathbb{X} \subset \mathbb{R}^d$. Тогда семейство линейных классификаторов $\mathbb{A}_h$ — это множество всех гиперплоскостей, разделяющих данную выборку:

$$\mathbb{A}_h = \Big\{ a_w(x) = \operatorname{sign}\big(\langle w, x\rangle + w_0\big) \mid w \in \mathbb{R}^d, w_0 \in \mathbb{R} \Big\}.$$

Как было сказано выше, множество алгоритмов мы будем отождествлять с множеством векторов ошибок этих алгоритмов. Это означает, что множество $\mathcal{A}_h$ мы будем отождествлять с множеством всех бинарных векторов, соответствующих разделению выборки на две части гиперплоскостью.

1.2 Представление семейства алгоритмов графом

Введем на множестве алгоритмов отношение частичного порядка $\prec$:

$$a \leqslant b \;\Leftrightarrow\; (\forall x \in \mathbb{X}\; I(a,x) \leqslant I(b,x)).$$

Если $a \leqslant b$ и при этом $\rho(a,b) = 1$ (здесь ρ — это хэммингово расстояние), то будем говорить, что a *предшествует* b и записывать $a \prec b$.

Определение 2. *Графом расслоения-связности семейства алгоритмов* $\mathbb{A}$ *называется ориентированный граф* $G = (V, E)$ *с множеством вершин* $V = \mathbb{A}$ *и множеством ребер* $E = \{(a,b) \mid a \prec b\}$.

Слоем графа расслоения-связности называется множество алгоритмов, допускающих одинаковое число ошибок: $A_m = \{a \in \mathcal{A} \mid n(a,\mathbb{X}) = m\}$. Граф расслоения-связности является многодольным, доли соответствуют слоям A_m, ребрами могут соединяться только соседние слои. В частности, из многодольности графа следует его двудольность.

Если две вершины графа a и b соединены ребром (т.е. $a \prec b$), то векторы ошибок алгоритмов a и b отличаются лишь в одном элементе. Это позволяет поставить в соответствие каждому ребру (a,b) объект $x_{ab} \in \mathbb{X}$, такой что $I(a,x_{ab}) = 0$ и $I(b,x_{ab}) = 1$.

ерхней полуокрестностью алгоритма a называется множество $C^+(a) = \{b \in \mathbb{A} \mid (a,b) \in E\}$. Аналогично, *нижней полуокрестностью* называется множество $C^-(a) = \{b \in \mathbb{A} \mid (b,a) \in E\}$. Элементы верхней и нижней полуокрестностей алгоритма a будем называть верхними и нижними соседями соответственно.

Вершина графа расслоения-связности называется *истоком*, если у нее нет входящих ребер.

Верхней связностью $u(a)$ алгоритма a называется число вершин в его верхней окрестности:

$$u(a) = |C_+(a)|.$$

Неполноценностью (inferiority) $q(a)$ алгоритма a называется число объектов $x \in \mathbb{X}$, на которых a ошибается, при том, что существует алгоритм $b \leqslant a$, не ошибающийся на x:

$$q(a) = \#\{x \in \mathbb{X} \mid I(a,x) = 1,\ \exists b \in \mathbb{A} :\ b \leqslant a, I(b,x) = 0\}.$$

Введем также обозначение для числа ошибок алгоритма a:

$$m(a) = n(a, \mathbb{X}).$$

2 Оценки вероятности переобучения

Классической оценкой вероятности переобучения является оценка Вапника-Червоненкиса (VC-оценка)

Теорема 1. *Для любой выборки $\mathbb{X}$, любого множества алгоритмов A, любого метода обучения μ и любого $\varepsilon \in [0,1]$ верна следующая оценка вероятности переобучения:*

$$Q_\varepsilon(\mu, \mathbb{X}) \leqslant \mathsf{P}\big[\max_{a \in A} \delta(a,X) \geqslant \varepsilon\big] \leqslant \sum_{a \in A} H_L^{\ell,m}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right), \quad m = n(a,\mathbb{X}). \tag{1}$$

Данная оценка является крайне завышенной и представляет лишь теоретический интерес.

Известна верхняя оценка вероятности переобучения, которая выражается через неполноценность и верхнюю связность всех алгоритмов множества $\mathbb{A}$ и справедлива для ПМЭР, значит, и для произвольного метода МЭР.

Теорема 2. *Для пессимистичного минимизатора эмпирического риска μ, любых $\mathbb{X}$, $\mathbb{A}$ и $\varepsilon \in (0,1)$*

$$Q_\varepsilon \leqslant \sum_{a\in\mathbb{A}} \frac{C_{L-u-q}^{\ell-u}}{C_L^\ell} \mathcal{H}_{L-u-q}^{\ell-u,\ m-q}\left(\tfrac{\ell}{L}(m-\varepsilon k)\right), \tag{2}$$

где $\mathcal{H}_L^{\ell,m}(z) = \sum_{s=0}^{\lfloor z\rfloor} \frac{C_m^s C_{L-m}^{\ell-s}}{C_L^\ell}$ — функция гипергеометрического распределения, $u \equiv u(a)$, $q \equiv q(a)$, $m \equiv n(a,\mathbb{X})$.

Данная оценка является точной для некоторых нетривиальных модельных семейств алгоритмов, но на реальных семействах оказывается завышенной на 1–2 порядка. В данной работе вводятся характеристики попарного сходства алгоритмов, которые позволяют улучшить данную оценку.

Определим для произвольных двух алгоритмов a_i и a_j множество A_{ij} объектов, на которых a_i не допускает ошибку, а a_j допускает:

$$A_{ij} = \{x \in \mathbb{X} \mid I(a_i,x) = 0,\ I(a_j,x) = 1\}.$$

Лемма 1. *Пусть алгоритмы пронумерованы в порядке неубывания числа ошибок, μ — ПМЭР. Тогда для произвольной обучающей выборки $X \subset \mathbb{X}$*

$$[\mu X = a_i] = \left(\prod_{j=1}^{i-1}\Big[|X\cap A_{ji}| \leqslant |X\cap A_{ij}|\Big]\right)\left(\prod_{j=i+1}^{D}\Big[|X\cap A_{ji}| < |X\cap A_{ij}|\Big]\right). \tag{3}$$

Доказательство. Заметим, что если $|X\cap A_{ji}| > |X\cap A_{ij}|$, то a_j допускает на обучающей выборке меньше ошибок, чем a_i. Значит, в этом случае a_i не может быть выбран методом μ.

Предположим теперь, что $|X\cap A_{ji}| = |X\cap A_{ij}|$ и $j > i$. Из равенства $|X\cap A_{ji}| = |X\cap A_{ij}|$ следует, что алгоритмы a_i и a_j допускают одинаковое число ошибок на обучении: $n(a_i,X) = n(a_j,X)$. Так как алгоритмы пронумерованы в порядке неубывания числа ошибок, то из $j > i$ следует $m(a_j) \geqslant m(a_i)$. Отсюда получаем:

$$n(a_j,\bar{X}) = m(a_j) - n(a_j,X) \geqslant m(a_i) - n(a_i,X) = n(a_i,\bar{X}).$$

Иными словами, a_i и a_j допускают одинаковое число ошибок на обучении, но при этом a_j допускает не меньше ошибок на контроле и имеет больший номер. Это означает, что a_i не может быть выбран методом обучения.

Итак, для того, чтобы алгоритм a_i был выбран ПМЭР μ на обучающей выборке X, необходимо, чтобы для любого $j = 1,\ldots,D$ было выполнено:

- $|X \cap A_{ji}| \leqslant |X \cap A_{ij}|$, если $j < i$
- $|X \cap A_{ji}| < |X \cap A_{ij}|$, если $j > i$

Значит, верно следующее неравенство:

$$[\mu X = a_i] \leqslant \left(\prod_{j=1}^{i-1} [|X \cap A_{ji}| \leqslant |X \cap A_{ij}|]\right) \left(\prod_{j=i+1}^{D} [|X \cap A_{ji}| < |X \cap A_{ij}|]\right)$$

Покажем, что данное неравенство выполнено и в другую сторону. Пусть правая часть равенства (3) равна единице. Рассмотрим произвольный алгоритм a_j, $i \neq j$. Если $|X \cap A_{ji}| < |X \cap A_{ij}|$, то a_j допускает на обучающей выборке больше ошибок, чем a_i, и поэтому не может быть выбран методом обучения μ. Если же $|X \cap A_{ji}| = |X \cap A_{ij}|$, то $j < i$, а значит $m(a_j) \leqslant m(a_i)$. Тогда $n(a_j, \bar{X}) \leqslant n(a_i, \bar{X})$, откуда следует, что алгоритм a_j не будет выбран методом обучения. Мы показали, что ни один алгоритм $a_j \in \mathbb{A}$, $i \neq j$ не будет выбран методом обучения. Значит, $\mu X = a_i$, и равенство единице правой части (3) является достаточным условием для выбора a_i, то есть

$$[\mu X = a_i] \geqslant \left(\prod_{j=1}^{i-1} [|X \cap A_{ji}| \leqslant |X \cap A_{ij}|]\right) \left(\prod_{j=i+1}^{D} [|X \cap A_{ji}| < |X \cap A_{ij}|]\right)$$

□

Непосредственное использование данного выражения для вычисления вероятности переобучения не представляется возможным из-за вычислительной сложности. Но его можно превратить в необходимое условие, оставив для алгоритма a_i только множители, соответствующие предшествующим ему истокам в графе расслоения-связности G и алгоритмам из его верхней полуокрестности. Нетрудно показать, что тогда из необходимого условия будет следовать комбинаторная оценка теоремы 2. В данной работе предлагается учесть связь алгоритма a_i с произвольным алгоритмом a_s.

Лемма 2. *Пусть μ — ПМЭР, a_i и a_s — два произвольных алгоритма из $\mathbb{A}$. Тогда*

$$\mathsf{P}\big[\mu X = a_i\big]\big[\nu(a_i, \bar{X}) - \nu(a_i, X) \geqslant \varepsilon\big] \leqslant \\ \leqslant \sum_{t=0}^{T_{is}} \frac{C_q^t C_{L-u-q}^{\ell-u-t}}{C_L^\ell} \mathcal{H}_{L-u-q}^{\ell-u-t,\, m-q}\left(\tfrac{\ell}{L}(m - \varepsilon k) - t\right),$$

где $u \equiv u(a_i)$, $q \equiv |A_{si}|$, $T_{is} = \min(|A_{is}|, |A_{si}|)$, $m \equiv n(a_i, \mathbb{X})$.

Доказательство. С помощью леммы 1 оценим величину $[\mu X = a_i]$ сверху, оставив только те множители, которые соответствуют a_s и верхней полуокрестности:

$$\begin{aligned}
[\mu X = a_i] &\leqslant ([s \leqslant i]\,[|A_{si} \cap X| \leqslant |A_{is} \cap X|] + [s > i]\,[|A_{si} \cap X| < |A_{is} \cap X|]) \times \\
&\quad \times \prod_{j\colon\, a_j \in C^+(a_i)} [|A_{ji} \cap X| < |A_{ij} \cap X|] \leqslant \\
&\leqslant [|A_{si} \cap X| \leqslant |A_{is} \cap X|] \prod_{j\colon\, a_j \in C^+(a_i)} [|A_{ji} \cap X| < |A_{ij} \cap X|] \leqslant \\
&\leqslant [|A_{si} \cap X| \leqslant |A_{is}|] \prod_{j\colon\, a_j \in C^+(a_i)} [|A_{ji} \cap X| < |A_{ij} \cap X|] = \\
&= [|A_{si} \cap X| \leqslant |A_{is}|] \prod_{j\colon\, a_j \in C^+(a_i)} [|A_{ij} \cap X| > 0]
\end{aligned}$$

Из последнего неравенства следует, что для того, чтобы был выбран алгоритм a_i, необходимо, чтобы в обучающую выборку попали все объекты из $\bigcup_{a_j \in C^+(a_i)} A_{ij}$, и чтобы из A_{si} в X попало не более $|A_{is}|$ объектов. Исходя из этих соображений, получаем следующую оценку:

$$\mathsf{P}[\mu X = a_i][\delta(a_i, X) \geqslant \varepsilon] \leqslant \sum_{t=0}^{\min(|A_{is}|, |A_{si}|)} \frac{C_{|A_{si}|}^{t} C_{L-u-|A_{si}|}^{\ell-u-t}}{C_L^{\ell}} \mathcal{H}_{L-u-|A_{si}|}^{\ell-u-t,\; m-|A_{si}|} \left(\frac{\ell}{L}(m - \varepsilon k) - t \right)$$

□

Распорядимся свободой выбора алгоритма a_s: для каждого a_i будем брать в качестве a_s тот из истоков графа расслоения-связности G, который даёт наименьший вклад в оценку.

Теорема 3. *Пусть μ — ПМЭР, S — множество всех истоков графа расслоения-связности. Тогда, в обозначениях леммы 2*

$$Q_\varepsilon \leqslant \sum_{i=1}^{D} \min_{s \in S} \left\{ \sum_{t=0}^{T_{is}} \frac{C_q^t C_{L-u-q}^{\ell-u-t}}{C_L^{\ell}} \mathcal{H}_{L-u-q}^{\ell-u-t,\, m-q} \left(\tfrac{\ell}{L}(m - \varepsilon k) - t \right) \right\}. \tag{4}$$

Доказательство. Распишем вероятность переобучения, используя формулу полной вероятности:

$$Q_\varepsilon(\mu, \mathbb{X}) = \mathsf{P}[\delta(\mu X, X) \geqslant \varepsilon] = \sum_{i=1}^{D} \mathsf{P}[\mu X = a_i][\delta(a_i, X) \geqslant \varepsilon]$$

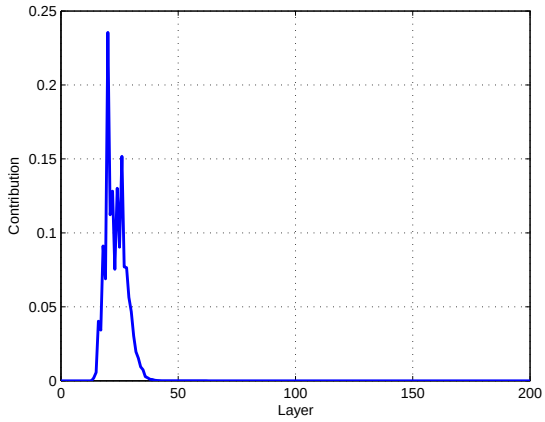


Рис. 1: Вклады слоев графа расслоения-связности в оценку (3). Оценка вычислена для семейства линейных классификаторов на линейно неразделимой двумерной выборке, состоящей из $L = 200$ объектов.

Для различных $a_s \in \mathcal{A}$ можно получить различные оценки для величины $\mathsf{P}[\mu X = a_i][\delta(a_i, X) \geqslant \varepsilon]$, используя лемму 2. Мы вычислим такие оценки для всех a_s, являющихся истоками графа расслоения-связности, а затем выберем наименьшую из таких оценок:

$$Q_\varepsilon(\mu, \mathbb{X}) = \sum_{i=1}^{D} \mathsf{P}[\mu X = a_i][\delta(a_i, X) \geqslant \varepsilon] \leqslant$$
$$\leqslant \sum_{i=1}^{D} \min_{s \in S} \left\{ \sum_{t=0}^{T_{is}} \frac{C^t_{|A_{si}|} C^{\ell-u-t}_{L-u-|A_{si}|}}{C^\ell_L} \mathcal{H}^{\ell-u-t,\ m-|A_{si}|}_{L-u-|A_{si}|} \left(\frac{\ell}{L}(m - \varepsilon k) - t \right) \right\}$$

□

Непосредственное вычисление оценки (4) требует перебора всех алгоритмов семейства $\mathbb{A}$, что на практике неосуществимо. Значимый вклад в оценку вносят лишь алгоритмы из некоторого числа t нижних слоев графа G (см. рис. 1). Тем не менее, даже этих алгоритмов может быть слишком много.

3 Обход графа расслоения-связности семейства линейных классификаторов

3.1 Необходимые сведения

3.1.1 Формула Шермана-Моррисона

Рассмотрим систему линейных уравнений $Ax = b$ с матрицей A размера $d \times d$. Допустим, нам известна обратная матрица A^{-1}.

Пусть наша система была изменена: $\tilde{A} = A + uv^T$. Тогда можно пересчитать обратную матрицу за $O(d^2)$ ([7]):

$$\tilde{A}^{-1} = (A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u} \tag{5}$$

Например, если $(A_1, \ldots, A_i, \ldots, A_d)$ — строки матрицы A, $(A_1, \ldots, \tilde{A}_i, \ldots, A_d)$ — строки $\tilde{A}$, то $\tilde{A} = A + uv^T$, где

$$u = (0, \ldots, 0, 1, 0, \ldots, 0)^T$$
$$v^T = \tilde{A}_i - A_i$$

(единица в u стоит в i-ой позиции).

3.1.2 Многогранники

Множество $P \subset \mathbb{R}^d$ называется *выпуклым многогранником*, если оно является пересечением конечного числа замкнутых полупространств. В дальнейшем мы будем опускать слово «выпуклый», поскольку никакие другие многогранники в данной работе не встречаются.

Гранью многогранника P — это либо сам P, либо его подмножество вида $P \cap h$, где h — такая гиперплоскость, что P целиком содержится в одном из определяемых ей полупространств.

Грани размерности 0 называется *вершинами*, размерности 1 — *ребрами*.

Следующая лемма описывает структуру граней многогранника.

Лемма 3 ([12]). *Пусть $F_1, \ldots, F_s$ — все грани размерности $(d-1)$ многогранника P. Тогда любая грань G размерности $i \leqslant d-1$ представима в виде пересечения*

некоторых граней размерности $(d-1)$*:*

$$G = F_{i_1} \cap \cdots \cap F_{i_{d-t}}$$

3.1.3 Конфигурации гиперплоскостей

Общее положение. Говоря «набор гиперплоскостей $\mathbb{H}$ находится в общем положении», мы будем иметь ввиду следующее:

- в $\mathbb{H}$ нет параллельных гиперплоскостей
- пересечение любых k гиперплоскостей $(2 \leqslant k \leqslant d)$ из $\mathbb{H}$ имеет размерность $d-k$

Под общим положением набора точек $\mathbb{X}$ будем подразумевать, что:

- никакие две точки из $\mathbb{X}$ не совпадают
- никакое подмножество из k точек $(k = 3, \ldots, d+1)$ не является аффинно зависимым (то есть не лежит в многообразии размерности $k-2$ или меньше)
- в $\mathbb{X}$ нет точки, лежащей в начале координат

Опишем несколько более общую точку зрения на понятие общего положения. Пусть $\mathcal{S}$ — множество объектов. Рассмотрим его подмножество $\mathcal{T}$, состоящее только из объектов, удовлетворяющих некоторым требованиям. Тогда, если мера множества $\mathcal{S} \setminus \mathcal{T}$ равна нулю, то эти требования называют *требованиями общего положения.*

Если какое-то утверждение удалось доказать для объектов, находящихся в общем положении, то, как правило, с помощью специальных техник удается обобщить доказательство на произвольный случай ([?]).

Преобразование двойственности. *Преобразование двойственности* $\mathcal{D}$ определяется следующим образом:

- произвольная точка $x \in \mathbb{R}^d$ переводится в гиперплоскость:

$$\mathcal{D}(x) = \{w \in \mathbb{R}^d \mid \langle w, x \rangle = 0\};$$

- Произвольная гиперплоскость $h = \{x \in \mathbb{R}^d \mid \langle w, x \rangle = 0\}$ переводится в точку:

$$\mathcal{D}(h) = w.$$

Видно, что если x — вектор из $\mathbb{R}^d$, то $\mathcal{D}(x)$ — это гиперплоскость с вектором нормали x, проходящая через начало координат. Заметим также, что $\mathcal{D}(x)$ состоит из нормалей ко всем гиперплоскостям, проходящим через точку x.

Пусть $\mathbb{X} = \{x_1, \ldots, x_L\} \subset \mathbb{R}^d$ — выборка в общем положении. Также разрешим наличие константного признака — в этом случае будем требовать, чтобы без данного признака выборка была в общем положении.

Переведем выборку в набор гиперплоскостей с помощью преобразования двойственности: $h_i = \mathcal{D}(x_i) = \{w \in \mathbb{R}^d \mid \langle w, x_i \rangle = 0\}$, $\mathbb{H} = \{h_1, \ldots, h_L\}$. Введем следующие обозначения для полупространств, образованных гиперплоскостью h_i: $h_i^+ = \{w \in \mathbb{R}^d \mid \text{sign}\langle w, x_i \rangle = y_i\}$, $h_i^- = \{w \in \mathbb{R}^d \mid \text{sign}\langle w, x_i \rangle = -y_i\}$. Заметим, что h_i^+ — это множество нормалей ко всем гиперплоскостям, правильно классифицирующим объект x_i, а h_i^- — множество нормалей ко всем гиперплоскостям, допускающим ошибку на x_i.

Теперь легко находить все гиперплоскости, соответствующие одному вектору ошибок. А именно, пусть дан вектор ошибок $(I_j)_{j=1}^L$, $I_j \in \{+, -\}$, где «+» означает, что классификатор дает правильный ответ на данном объекте, а «−» соответствует ошибке. Тогда множество всех линейных классификаторов, имеющих данный вектор ошибок, имеет вид $\bigcap_{j=1}^L h_j^{I_j}$.

Конфигурации гиперплоскостей Гиперплоскости из $\mathbb{H}$ разбивают $\mathbb{R}^d$ на выпуклые многогранники, называемые *ячейками конфигурации (cells of arrangement)*. Само же разбиение пространства на эти области называется *конфигурацией гиперплоскостей (arrangement of hyperplanes)*.

Следующая лемма показывает, как соотносятся выборка и двойственный ей набор гиперплоскостей.

Лемма 4. *Пусть $\mathbb{X}$ — выборка в общем положении, имеющая константный признак, $\mathbb{H}$ — двойственный ей набор гиперплоскостей, $\mathbb{A}_h$ — семейство линейных классификаторов для этой выборки.*

Тогда:

- *Существует взаимно однозначное соответствие между ячейками конфигурации гиперплоскостей $\mathbb{H}$ и алгоритмами из $\mathbb{A}_h$*

- *Если a_1 и a_2 — соседние алгоритмы, то соответствующие им ячейки C_{a_1} и C_{a_2} имеют общую грань размерности $(d-1)$*

Доказательство. Первое утверждение следует из приведенных выше рассуждений о построении множества всех линейных классификаторов, имеющих один и тот же вектор ошибок.

Покажем второе утверждение. Поскольку алгоритмы a_1 и a_2 соседние, их векторы ошибок отличаются ровно на одном объекте $x_{a_1 a_2}$. Уберем его из выборки; в этом случае алгоритмы a_1 и a_2 объединятся в один, поскольку их векторы ошибок будут совпадать. Вернем объект $x_{a_1 a_2}$ в выборку. В конфигурации, соответствующей выборке, появится новая гиперплоскость, которая будет проходить через ячейку, соответствующую объединению a_1 и a_2. Из требований общего положения (в частности, из аффинной независимости) вытекает, что эта гиперплоскость будет образовывать грань размерности $(d-1)$. Отсюда получаем второе утверждение леммы. □

3.2 Построение окрестности линейного классификатора

3.2.1 Переход к двойственной задаче

Итак, дан некоторый алгоритм $a_0 \in \mathbb{A}_h$, требуется найти все соседние алгоритмы. Будем считать, что a_0 задан вектором нормали z^0 к разделяющей гиперплоскости $h_0 = \{x \in \mathbb{R}^d \mid \langle z^0, x\rangle = 0\}$.

Будем считать, что константный признак имеет номер d.

Переведем нашу выборку в набор гиперплоскостей с помощью преобразования двойственности:

$$\mathbb{H} = \{h_i = \mathcal{D}(x_i) \mid i = 1, \ldots, L\}$$

Алгоритм a_0 будет переведен в точку: $\mathcal{D}(h_0) = z^0$.

Согласно лемме 4, ячейки конфигурации гиперплоскостей $\mathbb{H}$ взаимно однозначно соответствуют алгоритмам из $\mathbb{A}_h$, причем ячейки, соответствующие соседним алгоритмам, имеют общую грань размерности $(d-1)$.

3.2.2 Поиск вершины ячейки

Сначала нам нужно, имея точку внутри ячейки, найти какую-нибудь вершину этой ячейки.

Пусть $z^0 = (z_1^0, \ldots, z_d^0)$ — точка внутри ячейки C_a. Составим по ней следующую систему линейных уравнений:

$$\begin{cases} \langle w, x \rangle = c \\ \quad x_2 = z_2^0 \\ \quad \ldots \\ \quad x_d = z_d^0 \end{cases}$$

Здесь $\langle w, x \rangle = c$ — это уравнение некоторой гиперплоскости из $\mathbb{H}_N$. Выберем ту гиперплоскость, которая даст решение $(x_1, \ldots, x_d)$, максимально близкое к z^0, и обозначим его через z^1. Пусть это решение дает гиперплоскость $\langle w^1, x \rangle = c^1$.

Заметим, что z^1 принадлежит той же ячейке, что и z^0, но теперь уже не ее внутренности, а некоторой грани размерности $(d-1)$.

Перейдем к другой системе:

$$\begin{cases} \langle w^1, x \rangle = c^1 \\ \langle w, x \rangle = c \\ \quad x_3 = z_3^1 \\ \quad \ldots \\ \quad x_d = z_d^1 \end{cases}$$

Найдем гиперплоскость $\langle w^2, x \rangle = c^2$, дающую решение z^2, максимально близкое к z^1. Точка z^2 будет уже лежать на грани размерности $(d-2)$.

В конце концов мы получим точку z^d, удовлетворяющую системе

$$\begin{cases} \langle w^1, x \rangle = c^1 \\ \quad \ldots \\ \langle w^d, x \rangle = c^d \end{cases}$$

Эта точка и будет вершиной ячейки C_{a_0}.

Заметим, что при переходе от одной системы к другой меняется только одно уравнение и, возможно, правая часть. Значит, можно использовать формулу Шермана-Моррисона (5) для быстрого пересчета решения.

Время работы описанного метода имеет порядок $O(Nd^3)$.

3.2.3 Поиск всех вершин ячейки

Пусть v_0 — некоторая вершина ячейки C_{a_0}, лежащая на пересечении гиперплоскостей из множества $H_0 = \{h_{i_1}, \ldots, h_{i_d}\}$. Известно, что вершины и ребра выпуклого многогранника образуют связный граф ([6]). Отсюда следует, что можно найти все вершины многогранника, например, поиском в ширину из v_0. Для этого нам нужно уметь, находясь в некоторой вершине, находить все смежные с ней. Опишем, как это сделать на примере v_0.

Пусть вершины v и w являются пересечениями гиперплоскостей из множеств H_v и H_w соответственно, и пусть они соединены ребром (v, w). Ребро (v, w) является пересечением $(d-1)$-й гиперплоскости, вершины v и w — пересечением d гиперплоскостей. Значит, так как v и w лежат на одном и том же ребре (v, w), то множества H_v и H_w отличаются друг от друга только каким-то одним элементом (в оба обязательно должны входить гиперплоскости, образующие ребро).

Значит, чтобы найти все вершины, смежные с v_0, надо перебрать все возможные способы изменить один элемент в множестве H_0. Допустим, мы пробуем заменить h_{i_1} на другую гиперплоскость. Заметим, что пересечение гиперплоскостей $\{h_{i_2}, \ldots, h_{i_d}\}$ образует прямую, а точки из множества $V = \{v_j = h_j \cap h_{i_2} \cap \cdots \cap h_{i_d} \mid j = 1, \ldots, N, j \neq i_1\}$ лежат на этой прямой. Легко видеть, что вершина $v \in V$ будет смежной с v_0 только в том случае, если не найдется точки $v' \in V$, лежащей между v и v_0.

Пусть $v \in V$ — вершина, ближайшая к v_0 (отметим, что их может быть две — по одной с каждой стороны). Построим вектор ошибок соответствующего ей алгоритма. Так как эта точка лежит на d гиперплоскостях, то d позиций в ее векторе ошибок могут быть заполнены как угодно (каждый вариант заполнить их соответствует небольшому сдвигу во внутренность некоторой ячейки; в терминах прямой задачи это означает, что у нас есть гиперплоскость, проходящая через d точек, и мы можем задать любую классификацию на этих точках, немного двигая гиперплос-

кость). Если эти позиции могут быть заполнены так, что вектор ошибок совпадет с вектором ошибок a_0, то v является вершиной C_{a_0}, иначе — нет.

Отметим, что при замене какой-то одной гиперплоскости на другую точка пересечения может быть найдена за $O(d^2)$, так как в соответствующей СЛАУ меняется только одно уравнение.

Оценим сложность описанного метода. Пусть у ячейки s вершин. В каждой из вершин мы будем перебирать все варианты заменить одну гиперплоскость из ее описания ($O(Nd)$ способов), и для каждого такого способа искать точку пересечения d гиперплоскостей (применяем формулу Шермана–Моррисона, $O(d^2)$ операций). Значит, все вершины мы найдем за $O(sd^3N)$.

3.2.4 Поиск соседних алгоритмов

Пусть $\{v_0, v_1, \ldots, v_s\}$ — все вершины ячейки C.

Выберем v_0 и построим ее вектор ошибок. Как говорилось выше, d позиций в его векторе ошибок могут быть заполнены как угодно. Переберем все способы заполнить их таким образом, чтобы получился вектор, отличающийся от a_0 в одной позиции. Каждый такой способ будет соответствовать алгоритму, соседнему с a_0.

Проделав это со всеми вершинами, мы найдем всю окрестность a_0 (так как переберем все ячейки, пересекающиеся с C_{a_0}).

Это может быть проделано за $O(sd)$.

3.3 Приближенные методы построения окрестности [1]

Предложенный выше метод для построения окрестности линейного классификатора является точным, однако обладает рядом существенных недостатков. Во-первых, он работает лишь в том случае, когда выборка находится в общем положении; в реальных же задачах крайне часто встречаются дискретные или номинальные признаки, при наличии которых это требование нарушается. Во-вторых, он имеет большую сложность; так, его использование при отборе признаков (см. раздел 5) приводит к серьезным временным затратам. В данном разделе предлагается метод приближенного построения окрестности, лишенный этих недостатков.

[1] Работа, представленная в данном разделе, проделана совместно с Александром Фреем

Идея метода заключается в том, что провести из точки двойственного пространства, соответствующей линейному классификатору, несколько лучей, и найти все ячейки, которые этими лучами пересекаются; затем из найденных ячеек выделяются соседи данного классификатора. Опишем метод подробнее.

3.3.1 Поиск вдоль луча

Пусть a_0 — линейный классификатор, заданный вектором нормали w_0. В двойственном пространстве этот классификатор соответствует точке w_0, расположенной в ячейке C_{a_0}.

Пусть $v \in \mathbb{R}^d$ — произвольный вектор. Рассмотрим луч, исходящий из точки w_0, направление которого задается вектором v:

$$w = w_0 + tv, \qquad t \geqslant 0.$$

Найдем пересечения этого луча со всеми гиперплоскостями из $\mathbb{H}$. В точке пересечения с гиперплоскостью h_i выполнено $\langle w_0 + tv, x_i \rangle$. Отсюда находим, что пересечение происходит в точке t_i:

$$t_i = -\frac{\langle w_0, x_i \rangle}{\langle v, x_i \rangle}.$$

Отсортируем все точки пересечения по возрастанию:

$$t_{(1)} < t_{(2)} < \cdots < t_{(L)}.$$

Точка $t_{(1)}$ соответствует пересечению луча с границей многогранника C_{a_0}. Если пересечение находится на грани максимальной размерности (т.е. $d-1$), то дальше луч проходит через ячейку, соответствующую соседнему с a_0 классификатору; в противном случае может оказаться, что ячейка соседней не является. Для того, чтобы проверить, является ли следующая ячейка соседней, достаточно взять классификатор с вектором нормали w', лежащим на луче между точками $t_{(1)}$ и $t_{(2)}$:

$$w' = \frac{1}{2}\Big(\big(w_0 + t_{(1)}v\big) + \big(w_0 + t_{(2)}v\big)\Big),$$

и проверить, отличается ли его вектор ошибок от w_0 лишь на одном объекте.

Таким образом, мы научились находить соседний классификатор вдоль заданного направления.

3.3.2 Построение случайных направлений

Опишем теперь возможные способы построения направлений, вдоль которых будет осуществляться поиск. Наиболее простым подходом является генерация каждой компоненты v_i вектора v из равномерного распределения на $[0, 1]$.

Отметим, что существуют методы, позволяющие расположить заданное число точек на сфере так, что расстояния между ними будут максимальны [18]. Такой подход представляется логичным, поскольку позволил бы равномерно исследовать все пространство вокруг текущей ячейки. Однако такие методы имеют достаточно большую сложность (так, в работе [18] предлагается алгоритм со сложностью порядка $O(d^{1.9}N^{0.6})$, где d — размерность пространства, а N — число точек), в то время как перед нами стоит задача уменьшения времени работы.

В дальнейшем мы будем пользоваться простым способ с генерацией вектора из равномерного распределения.

4 Случайные блуждания и их применение для вычисления оценок

Пусть $G = (V, E)$ — неориентированный граф, на вершинах которого задана функция $f : V \to \mathbb{R}$, и требуется вычислить ее матожидание относительно некоторого распределения $\boldsymbol{p} = (p(v))_{v \in V}$ на вершинах:

$$F = \mathbb{E}_{\boldsymbol{p}} f = \sum_{v \in V} f(v)p(v). \tag{6}$$

Нашей задачей будет приближенное вычисление данного матожидания.

4.1 Независимое сэмплирование

Предположим, что нам известно множество всех вершин V. Сгенерируем выборку из n вершин $v_1, \ldots, v_n$ так, что вероятность выбора вершины v равна $p(v)$. Тогда искомое матожидание (6) можно оценить следующим образом:

$$\chi(f) = \frac{1}{n} \sum_{i=1}^{n} f(v_i). \tag{7}$$

Данная оценка является несмещенной:

$$\mathbb{E}\chi(f) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}f(v_i) = \frac{1}{n}\sum_{i=1}^{n}\sum_{v\in V} f(v)p(v) = \frac{1}{n}\sum_{i=1}^{n} F = F.$$

Опишем несколько более общий подход для случая, когда неудобно генерировать выборку непосредственно из распределения $\boldsymbol{p}$. Пусть выборка $v_1, \ldots, v_n$ сгенерирована так, что вероятность выбора вершины v равна $q(v)$, где $\boldsymbol{q} = (q(v))_{v\in V}$ — произвольное распределение. В этом случае предлагается использовать оценку

$$\chi(f) = \frac{1}{n}\sum_{i=1}^{n} f(v_i)\frac{p(v_i)}{q(v_i)}. \tag{8}$$

Данная оценка также является несмещенной:

$$\mathbb{E}\chi(f) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}f(v_i)\frac{p(v_i)}{q(v_i)} = \frac{1}{n}\sum_{i=1}^{n}\sum_{v\in V} f(v)\frac{p(v)}{q(v)}q(v) = \frac{1}{n}\sum_{i=1}^{n}\sum_{v\in V} f(v)p(v) = \frac{1}{n}\sum_{i=1}^{n} F = F.$$

В случае, когда $\boldsymbol{p}$ является равномерным распределением, оценка (8) называется оценкой Хансена-Гурвица.

4.2 Сэмплирование с помощью случайных блужданий

Оценки, основанные на независимом сэмплировании, являются точными и удобными для вычисления, однако для их использования нужно уметь генерировать с определенными вероятностями вершины из множества V. Зачастую это оказывается невозможным, и поэтому приходится прибегать к техникам, основанным на обходе графа.

Различные техники обхода графа формализуются с помощью теории *марковских цепей*. Пусть $V_1, V_2, \ldots$ — случайные величины, принимающие значения из множества V вершин графа G.

Определение 3. *Последовательность случайных величин* $\{V_t \in V \,|\, t = 0, 1, 2, \ldots\}$ *называется марковской цепью, если*

$$\mathbb{P}(V_{t+1} = v \,|\, V_1 = v_1, \ldots, V_t = v_t) = \mathbb{P}(V_{t+1} = v \,|\, V_t = v_t).$$

Нас будут интересовать *стационарные* марковские цепи, то есть такие, что

$$\mathbb{P}(V_{t+1} = v \,|\, V_t = u) = \mathbb{P}(V_t = v \,|\, V_{t-1} = u).$$

Так как значение марковской цепи на $(t+1)$-м шаге зависит только от t-го шага, то ее можно полностью охарактеризовать *матрицей перехода* $P = (p_{vu})_{v,u \in V}$:

$$p_{vu} = \mathbb{P}(V_{t+1} = v \,|\, V_t = u)$$

(здесь мы также воспользовались стационарностью — без нее p_{vu} зависело бы от индекса t).

Дадим интерпретацию марковской цепи. Допустим, мы стартуем из некоторой вершины v_0 и начинаем блуждать по графу, каждый раз случайно выбирая новую вершину. Тогда значение случайной величины V_t — это вершина, в которой мы находимся в момент времени t, а p_{vu} — вероятность, с которой мы переходим из вершины v в вершину u. Будем требовать, чтобы переходы были возможны лишь по ребрам графа: $(v, u) \notin E \Rightarrow p_{vu} = 0$.

Марковская цепь называется *неприводимой*, если из любой ее вершины можно попасть в любую с ненулевой вероятностью:

$$\forall v, u \in V \ \exists s: \ \mathbb{P}(V_s = v \,|\, V_0 = u) > 0.$$

Цепь называется апериодической, если для всех вершин v выполнено

$$k_v = \gcd\{t \,|\, \mathbb{P}(V_t = v \,|\, V_0 = v) > 0\} = 1$$

Если же для некоторой вершины $k_v \geqslant 2$, то говорят, что вершина v имеет период k. Это означает, что, выйдя из вершины, мы можем вернуться в нее лишь через число шагов, кратное k_v.

Распределение $\boldsymbol{\pi}$ на вершинах графа называется *стационарным*, если

$$P^T \pi = \pi.$$

Обсудим определение подробнее. Допустим, вероятность попасть в вершину v на t-м шаге равна $\pi(v)$. Тогда вероятность попасть в вершину u на $(t+1)$ шаге вычисляется как

$$\mathbb{P}(V_{t+1} = u) = \sum_{v \in V} \pi(v) p_{vu} = \pi(u).$$

Таким образом, если на некотором шаге распределение вершин оказалось равным $\boldsymbol{\pi}$, то оно и останется таким же на всех последующих шагах.

Далее мы будем предполагать, что цепь $\{V_t\}$ является неприводимой и апериодической. В этом случае цепь всегда имеет стационарное распределение $\boldsymbol{\pi}$.

Напомним, что нашей основной целью является приближенное вычисление матожидания $\mathbb{E}_{\boldsymbol{\pi}} f$. Определим следующую оценку для него:

$$\mu_n(f) = \frac{1}{n} \sum_{i=1}^{n} f(V_i).$$

Следующая теорема показывает, что данная оценка является состоятельной.

Теорема 4 (Закон больших чисел для марковских цепей)**.** *Пусть $\{V_t\}$ — неприводимая марковская цепь со стационарным распределением $\boldsymbol{\pi}$. Тогда для любого начального распределения $\mathbb{P}(V_0 = v)$ и для любой функции $f : V \to \mathbb{R}$, такой что $\mathbb{E}_{\boldsymbol{\pi}}|f| < \infty$, выполнено*

$$\mu_n(f) \xrightarrow[n\to\infty]{\text{a. s.}} \mathbb{E}_{\boldsymbol{\pi}} f.$$

Замечание 1. *Из данной теоремы следует, что оценка $\mu_n(f)$ является асимптотически несмещенной для $\mathbb{E}_{\boldsymbol{\pi}} f$.*

Последовательность $\{\mu_n(f)\}$ ограничена почти всюду:

$$|\mu_n(f)| = \left| \frac{1}{n} \sum_{i=1}^{n} f(V_i) \right| \leqslant \max_{v \in V} |f(v)|.$$

По теореме об ограниченной сходимости [11, p. 180] из ограниченности и из сходимости последовательности почти всюду следует сходимость матожиданий

$$\mathbb{E}\mu_n(f) \xrightarrow[n\to\infty]{} \mathbb{E}_{\boldsymbol{\pi}} f,$$

что и означает асимптотическую несмещенность.

Итак, $\mu_n(f)$ является несмещенной и состоятельной оценкой для матожидания функции f по стационарному распределению $\boldsymbol{\pi}$. Опишем теперь, как вычислить матожидание по произвольному распределению $\boldsymbol{p}$. Зададим корректирующую функцию

$$w(v) = \frac{p(v)}{\pi(v)}$$

и построим оценку

$$\mu_n(wf) = \frac{1}{n} \sum_{i=1}^{n} w(V_i) f(V_i). \tag{9}$$

Тогда, по теореме 4

$$\mu_n(wf) = \frac{1}{n}\sum_{i=1}^{n} w(V_i)f(V_i) \xrightarrow[n\to\infty]{\text{a. s.}} \mathbb{E}_{\boldsymbol{\pi}} wf = \\ = \sum_{v\in V} f(v)\frac{p(v)}{\pi(v)}\pi(v) = \sum_{v\in V} f(v)p(v) = \mathbb{E}_{\boldsymbol{p}} f,$$

то есть оценка (9) является состоятельной для $\mathbb{E}_{\boldsymbol{p}} f$. Также, согласно замечанию 1, она является асимптотически несмещенной.

Рассмотрим конкретный случай, где требуемое распределение $\boldsymbol{p}$ на вершинах является равномерным, а переход из текущей вершины в одну из соседних осуществляется равновероятно:

$$p_{vu} = \begin{cases} \frac{1}{d(v)}, & \text{если } (v,u)\in E; \\ 0, & \text{иначе.} \end{cases}$$

Такое случайное блуждание будем называть *простым*. Известно, что матрица перехода $P = (p_{vu})_{v,u\in V}$ задает неприводимую марковскую цепь с единственным стационарным распределением $\boldsymbol{\pi}$.

Лемма 5. *Распределение* $\pi(v) = \frac{\deg(v)}{2|E|}$ *является стационарным для простого случайного блуждания.*

Доказательство.

$$\sum_{v\in V}\pi(v)p_{vu} = \sum_{(v,u)\in E}\frac{\deg(v)}{2|E|}\frac{1}{\deg(v)} = \sum_{(v,u)\in E}\frac{1}{2|E|} = \frac{\deg(u)}{2|E|} = \pi(u).$$

□

Весовая функция для простого случайного блуждания принимает вид

$$w(v) = \frac{p(v)}{\pi(v)} = \frac{2|E|}{|V|}\frac{1}{\deg(v)}$$

Непосредственное использование оценки (9) для простого случайного блуждания зачастую оказывается непрактичным, поскольку в весовой функции используются величины $|V|$ и $|E|$. Точное значение этих величин на практике обычно неизвестно. Для решения проблемы рассмотрим оценку

$$\hat{F}_{\text{RW}} = \frac{\mu_n(wf)}{\mu_n(w)} = \frac{\sum_{i=1}^{n} w(V_i)f(V_i)}{\sum_{i=1}^{n} w(V_i)}. \tag{10}$$

В ней величины $|E|$ и $|V|$ выносятся за знак суммы и в числителе, и в знаменателе, и сокращаются:

$$\hat{F}_{\mathrm{RW}} = \frac{\mu_n(wf)}{\mu_n(w)} = \frac{\sum_{i=1}^{n} \frac{2|E|}{|V|} \frac{1}{\deg(V_i)} f(V_i)}{\sum_{i=1}^{n} \frac{2|E|}{|V|} \frac{1}{\deg(V_i)}} = \frac{\sum_{i=1}^{n} f(V_i)/\deg(V_i)}{\sum_{i=1}^{n} 1/\deg(V_i)}.$$

Сохраняются ли свойства оценки (9) после такого трюка? По теореме 4:

$$\mu_n(wf) \xrightarrow[n\to\infty]{\text{a. s.}} \mathbb{E}_{\boldsymbol{p}} f,$$

$$\mu_n(w) \xrightarrow[n\to\infty]{\text{a. s.}} \mathbb{E}_{\boldsymbol{\pi}} w = \sum_{v\in V} \frac{p(v)}{\pi(v)} \pi(v) = \sum_{v\in V} p(v) = 1$$

Отсюда следует [2], что $\hat{F}_{\mathrm{RW}} \xrightarrow[n\to\infty]{\text{a. s.}} \mathbb{E}_{\boldsymbol{p}} f$, т.е. оценка (10) является состоятельной для F.

4.2.1 Послойное вычисление оценки

Опишем еще один способ оценивания величины $F = \mathbb{E}_{\boldsymbol{p}} f$, который, согласно экспериментам, позволяет уменьшить дисперсию итоговых оценок.

Рассмотрим случай простого случайного блуждания, где $p(v) = 1/|V|$ — равномерное распределение на вершинах. Разобьем множество вершин графа G на m непересекающихся подмножеств $C_1, \ldots, C_m$ [3]. Обозначим их мощности через $c_1, \ldots, c_m$. Перепишем оцениваемое матожидание F в следующем виде:

$$F = \mathbb{E}_{\boldsymbol{p}} f = \frac{1}{|V|} \sum_{v\in V} f(v) = \frac{1}{|V|} \sum_{i=1}^{m} \left(\frac{1}{c_i} \sum_{v\in C_i} f(v) \right) c_i.$$

Отсюда можно получить следующую оценку для F:

$$\hat{F}'_{\mathrm{RW}} = \frac{1}{|V|} \sum_{i=1}^{m} \hat{f}_i \hat{c}_i, \tag{11}$$

где f_i — среднее значение функции f по множеству C_i.

Величину f_i предлагается оценивать по формуле (10), а мощность подмножества c_i — по следующей формуле (см. [26]):

$$\hat{c}_i = \frac{|V|}{n} \sum_{j=1}^{n} \left(\frac{1}{\deg(V_j)} \sum_{u\in N(V_j)} [u \in C_i] \right). \tag{12}$$

Отметим, что величина $|V|$ сокращается при подстановке (12) в (11), поэтому нам по-прежнему не нужно знать число вершин и ребер во всем графе G.

[2] Если $X_n \xrightarrow[n\to\infty]{\text{a. s.}} X$ и $Y_n \xrightarrow[n\to\infty]{\text{a. s.}} Y$, и при этом Y почти всюду отлично от нуля, то $X_n/Y_n \xrightarrow[n\to\infty]{\text{a. s.}} X/Y$.

[3] В случае с графом расслоения-связности это могут быть, например, его слои.

4.3 Характеристики марковских цепей

Существует много различных модификаций простого случайного блуждания, и возникает вопрос о том, как их следует сравнивать. В литературе особое внимание уделяется двум критериям: скорости перемешивания и асимптотической дисперсии.

4.3.1 Скорость перемешивания

Зафиксируем стартовую вершину случайного блуждания v_0. В этом случае можно вести речь о распределении марковской цепи на i-м шаге $\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}} = \left(p^i\left(v_0, v\right)\right)_{v\in V}$, $p^i(v_0, v) = \mathbb{P}(V_i = v \,|\, V_0 = v_0)$. Если начальное распределение отличается от стационарного, то на первых шагах распределение $\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}}$ также не совпадает со стационарным. Говоря неформально, скорость, с которой распределение $\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}}$ сходится к стационарному, и называется скоростью смешивания.

Основные результаты о сходимости распределения на i-м шаге к стационарному распределению доказаны для так называемого вариационного расстояния. *Вариационным расстоянием (total variation distance)* между распределениями $\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}}$ и $\boldsymbol{\pi}$ называется величина [4]

$$\|\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}} - \boldsymbol{\pi}\|_{\mathrm{TV}} = \max_{A\subset V} |p^i_{v_0}(A) - \pi(A)| = \frac{1}{2}\sum_{v\in V} |p^i_{v_0}(v) - \pi(v)|.$$

Следующая теорема утверждает, что распределение цепи на i-м шаге сходится к стационарному в смысле вариационного расстояния.

Теорема 5 ([21]). *Пусть $\{V_t\}$ — конечная, неприводимая и апериодическая марковская цепь со стационарным распределением $\boldsymbol{\pi}$. Тогда для любого начального распределения $\mathbb{P}(X_0 = v)$ выполнено*

$$\|\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}} - \boldsymbol{\pi}\|^2_{\mathrm{TV}} \xrightarrow[n\to\infty]{} 0.$$

[4] Существует связь между вариационным расстоянием и дивергенцией Кульбака-Лейблера, устанавливаемая неравенством Пинскера:

$$\mathrm{KL}(P\|Q) \geqslant \frac{1}{2}\|P - Q\|^2_{\mathrm{TV}}$$

Замечание 2. *Обратим внимание на тот факт, что в данной теореме требуется апериодичность марковской цепи. Если цепь имеет период, то ее распределение не сходится к стационарному, хотя теорема 4 (закон больших чисел) по-прежнему выполняется, и оценки матожиданий по-прежнему имеют смысл.*

Известна следующая оценка на вариационное расстояние для простого случайного блуждания [9]:

$$\|\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}} - \boldsymbol{\pi}\|^2_{\mathrm{TV}} \leqslant C\lambda_2^{2i}, \tag{13}$$

где C — некоторая константа, а λ_2 — второе по величине собственное значение матрицы перехода P [5].

Скорость перемешивания определяется как

$$t_{\mathrm{mix}}(\varepsilon) = \min\{i \geqslant 1 \mid \|\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{v_0}} - \boldsymbol{\pi}\|_{\mathrm{TV}} \leqslant \varepsilon\}.$$

Из оценки (13) следует, что время перемешивания можно уменьшить, уменьшив второе собственное значение матрицы перехода. Существует неравенство, формализующее эту мысль [22]:

$$t_{\mathrm{mix}}(\varepsilon) \leqslant \frac{1}{1-\lambda_2}\left(\ln\frac{1}{\pi(v_0)} + \ln\frac{1}{\varepsilon}\right).$$

Учет сэмплов, полученных до сходимости цепи к стационарному распределению, в оценке (9), может привести к смещенности оценки [5]. Значит, можно пытаться улучшить качество оценивания путем уменьшения собственного значения λ_2. Однако, экспериментально было показано, что даже после того, как марковская цепь достигнет стационарного состояния, все равно требуется много сэмплов для точного оценивания величины F, поэтому скорость смешивания может оказаться не самым лучшим критерием для сравнения методов [17].

[5] Пусть D —диагональная матрица, где $d_{ii} = \deg(i)$. Нормализуем матрицу перехода P: $P' = D^{-1/2}PD^{1/2}$. Полученная нормализованная матрица обладает рядом интересных свойств. Ее наименьшее собственное значение равно минус единице, если граф двудольный; второе по величине собственное значение меньше единицы тогда и только тогда, когда граф связный; кратность нулевого собственного значения равна числу компонент связности.

4.3.2 Асимптотическая дисперсия

Для марковских цепей имеется аналог центральной предельной теоремы.

Теорема 6 (ЦПТ). *Пусть $\{V_t\}$ — неприводимая марковская цепь со стационарным распределением $\boldsymbol{\pi}$. Тогда для любого начального распределения $\mathbb{P}(X_0 = v)$ и для любой функции $f : V \to \mathbb{R}$, такой что $\mathbb{E}_{\boldsymbol{\pi}} f^2 < \infty$, выполнено*

$$\sqrt{n}\,(\mu_n(f) - \mathbb{E}_{\boldsymbol{\pi}} f) \underset{n\to\infty}{\overset{d}{\Longrightarrow}} \mathcal{N}(0, \sigma^2(f)),$$

где $\sigma^2(f)$ называется асимптотической дисперсией оценки $\mu_n(f)$.

Теорема 6, что оценка (9) (а также (10) [6]) является асимптотически нормальной с асимптотической дисперсией $\sigma^2(f)$. Значит, что для достаточно больших n

$$\mathbb{P}\left\{\frac{\mu_n(f) - \mathbb{E}_{\boldsymbol{\pi}}(f)}{\sigma(f)/\sqrt{n}} > x\right\} \approx \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{y^2}{2}} dy,$$

откуда можно получить доверительный интервал для $\mathbb{E}_{\boldsymbol{\pi}} f$. Например, для достаточно больших n, с вероятностью не менее 0.95 выполнено

$$\mathbb{E}_{\boldsymbol{\pi}} f \in \left[\mu_n(f) - 2\frac{\sigma(f)}{\sqrt{n}}, \mu_n(f) + 2\frac{\sigma(f)}{\sqrt{n}}\right].$$

Видно, что повысить точность оценки можно двумя способами: увеличив размер выборки n или уменьшив асимптотическую дисперсию $\sigma^2(f)$. При это при равных размерах выборки более точные оценки будет давать метод с меньшей дисперсией.

4.4 Модификации простого случайного блуждания

4.4.1 Ленивое случайное блуждание

Как говорилось выше, распределение марковской цепи сходится к стационарному только при условии, что цепь является апериодической. Однако, любую цепь можно модифицировать так, что эта сходимость будет иметь место в любом случае. А именно, на каждом шаге с вероятностью $1/2$ будем оставаться в данной вершине, и с вероятностью $1/2$ будем делать обычный шаг случайного блуждания.

[6] Теорема Слуцкого [11, p. 318] гласит, что если $X_n \overset{d}{\Longrightarrow} X$ и $Y_n \overset{\mathbb{P}}{\Longrightarrow} c$, и при этом $c \neq 0$, то $X_n/Y_n \overset{d}{\Longrightarrow} X/c$. В нашем случае $X_n = \sqrt{n}\,(\mu_n(f) - \mathbb{E}_{\boldsymbol{\pi}} f)$, $Y_n = \mu_n(f)$, $c = 1$.

Algorithm 4.1 Frontier sampling

Вход: Граф $G = (V, E)$, набор стартовых вершин $P = (v^1, \ldots, v^s)$, число итераций i_{max}

Выход: Выборка $v_1, v_2, \ldots, v_{i_{max}}$

1: **для** $i = 1, \ldots, i_{max}$

2: Выбрать вершину $v \in P$ с вероятностью $\frac{deg(v)}{\sum_{u \in P} deg(u)}$

3: $R := \{v' \in V \mid (v, v') \in E\}$ // *окрестность вершины* v

4: Выбрать случайно вершину v' из равномерного распределения на R

5: $v_i := v'$

6: Заменить в P вершину v на v'

Пусть P — матрица перехода цепи, π — ее стационарное распределение. Описанная модификация соответствует переходу к матрице $P_{\text{lazy}} = (P + I)/2$. Для нее выполнено равенство $P_{\text{lazy}}^T \pi = (P^T \pi + \pi)/2 = \pi$, то есть после модификации π остается стационарным распределением. Более того, из любой вершины мы можем попасть в нее же саму за один шаг, поэтому новая цепь является апериодической, и для нее верна теорема 5, то есть имеет место сходимость к стационарному распределению.

4.4.2 Множественное случайное блуждание

Если граф состоит из нескольких слабо связанных между собой подграфов, то простое случайное блуждание может застрять в одном из них, что, в свою очередь, может привести к неточным оценкам. С точки зрения описанных выше характеристик марковских цепей, в данном случае имеет место крайне низкая скорость перемешивания — если в графе есть сильно обособленные компоненты, то второе собственное значение матрицы перехода оказывается крайне близким к единице, что приводит к низкой скорости сходимости в (13).

Наиболее простое и очевидное решение данной проблемы — запуск нескольких независимых случайных блужданий, объединение полученных ими выборок и вычисление оценки по такой совокупной выборке. Однако, оказывается, что такой подход далеко не всегда улучшает качество оценки, а зачастую даже ухудшает его [20]. Проблема заключается в том, что в данном подходе стационарное распределение от-

личается от стационарного распределения простого случайного блуждания (более того, неизвестен его вид), поэтому оценка (10) перестает быть состоятельной.

Стационарное распределение можно сохранить при использовании нескольких случайных блужданий, если сделать их *зависимыми*, что и предлагается в методе Frontier Sampling (FS) [20], см. алгоритм 4.1. В нем на каждом шаге с определенными вероятностями выбирается одно из s случайных блужданий, и в этом блуждании делается один шаг. Вероятности подобраны так, чтобы стационарное распределение совпадало со стационарным распределением простого случайного блуждания.

4.4.3 Случайное блуждание с перезапусками

Как говорилось выше, скорость перемешивания можно уменьшить, уменьшив второе собственное значение λ_2 матрицы перехода P. Метод RW-ur (random walk with uniform restarts) [5] направлен как раз на уменьшение λ_2.

Второе собственное значение графа характеризует его связность. Если граф несвязный, то оно равно единице; если в графе есть несколько слабо связанных компонент, то оно очень близко к единице. Значит, для уменьшения λ_2 нужно увеличить связность графа, добавив в него дополнительные ребра. В методе RW-ur предлагается соединить ребрами все пары вершин, приписав новым ребрам небольшие одинаковые вероятности. С точки зрения случайного блуждания это означает, что на каждом шаге с небольшой вероятностью α мы будем переходить в случайную вершину, равномерно выбранную из V. В этом случае стационарное распределение имеет вид

$$\pi(v) = \frac{\deg(v) + \alpha}{2|E| + \alpha|V|}.$$

Показано, что такая модификация действительно уменьшает второе собственное значение:

$$\hat{\lambda}_2 \approx \left(1 - \frac{\alpha}{\bar{\deg}}\right) \lambda_2.$$

4.4.4 Случайное блуждание без возвращений

Опишем метод RW-nb (non-backtracking random walk) [17], направленный на уменьшение асимптотической дисперсии.

Рассмотрим t-й шаг данного метода. Предположим, что сейчас мы находимся в вершине v_t, и пришли в нее мы из вершины v_{t-1}. Тогда вершину v_{t+1} предлагается выбирать равновероятно из всех соседей v_t за исключением вершины v_{t-1}. Иными словами, мы запрещаем случайному блужданию возвращаться назад.

Показано, что асимптотическая дисперсия данного метода *не превосходит* асимптотическую дисперсию простого случайного блуждания. В экспериментах же оценки получаются гораздо более точными по сравнению с оценками простого блуждания (так, для достижения того же качества оценивания ему требуется на 30-60% меньше сэмплов), что говорит о значительном улучшении асимптотической дисперсии.

4.5 Вычисление оценок вероятности переобучения с помощью случайных блужданий

Опишем теперь схему вычисления комбинаторных оценок переобучения для семейств линейных классификаторов, хорошо зарекомендовавшую себя в экспериментах (см. главу 5).

Сначала строится линейный классификатор методом SVM. Ему соответствует некоторая вершина в графе расслоения-связности G. Из этой вершины осуществляется спуск вниз по графу до истока. Затем из этого истока запускается обход всех слоев графа вплоть до (m_0+3)-го, где m_0 — число ошибок найденного истока, и фиксируются множество найденных истоков S.

Мы будем пользоваться оценкой (11), которая приближает средний вклад алгоритма, в то время как нас интересует сумма вкладов алгоритмов. Поэтому нам необходимо оценить $|V_t|$ — число вершин в нижних t слоях графа расслоения-связности, которые мы будем обозначать через $G_t = (V_t, E_t)$. Оценить $|V_t|$ достаточно просто: сгенерируем некоторое число линейных классификаторов (мы будем использовать 20000), и посчитаем, какая их доля p лежит в нижних t слоях. Поскольку нам известно число вершин $|V|$ во всем графе (оно равно $2\sum_{k=0}^{d-1} C_{L-1}^k$), то можно оценить $|V_t|$ как $p|V|$.

Далее генерируется 2000 классификаторов из $t = m_0 + 20$ нижних слоев графа с помощью случайного блуждания методом Frontier Sampling, в котором в качестве

множества стартовых вершин берется множество найденных истоков S. По полученной выборке оцениваются мощности слоев по формуле (12), и затем вычисляется приближённая оценка (11).

4.6 Эвристический подход к сэмплированию [7]

Вычисление комбинаторных оценок вероятности переобучения с помощью случайных блужданий осуществляется в два этапа: сначала осуществляется обход графа по определенной схеме, в результате чего получается выборка вершин; затем по данной выборке оценивается интересующая нас оценка вероятности переобучения. Оценивание производится с учетом свойств вершин из выборки, влияющих на их вероятность быть выбранными в процессе обхода графа. Благодаря этому оценки получаются состоятельными, то есть при увеличении длины выборки к бесконечности можно ожидать, что они будут все ближе к истинному значению.

Тем не менее, в экспериментах оказывается, что для достижения высокой точности оценивания на больших графах необходимы выборки размером в сотни тысяч вершин [20, 17]. В то же время, из-за высокой сложности поиска окрестности вершины, мы можем позволить себе сэмплирование не более чем тысячи вершин [8]. Более того, необходимо оценивать мощность нижних слоев, искать истоки графа, и решать прочие проблемы, требующий значительных вычислительных затрат.

Итак, точность оценки случайного блуждания никак не гарантируется при небольшом объеме выборки, но при этом оценивание требует существенных вычислительных затрат, сильно затрудняющих практическое применение. В данном разделе мы упростим задачу и предложим эвристический подход к ее решению, основанный на идеях случайных блужданий.

[7] Метод, описанный в данном разделе, предложен Александром Фреем

[8] Отметим, что здесь необходимо полностью находить окрестность вершины, поскольку в оценках используются степени вершин. Если бы эта информация не требовалась, то можно было бы использовать приближенный поиск соседей с помощью случайных направлений, описанный в разделе 3.3.

4.6.1 Приближенное вычисление оценок для отбора признаков

Одним из наиболее перспективных применений комбинаторных оценок является отбор признаков [9] В этом случае оценки вычисляются по одному и тому же набору объектов, но по разным подмножествам признаков, и выступают в качестве критерия качества признаковых подпространств. В данном случае центральным требованием к способу приближенного вычисления оценок является положительная коррелированность приближений с истинными оценками. Если это требование выполнено, то подмножество признаков, оптимальное с точки зрения комбинаторных оценок, будет оптимальным и с точки зрения их приближений, и поэтому замена оценок на их приближения не отразится на процедуре отбора признаков.

Также примем во внимание, что графы расслоения-связности реальных семейств имеют крайне большой размер, и поэтому мы не можем вычислить для них истинную оценку вероятности переобучения. Значит, требовать коррелированности приближений оценок с их истинными значениями не имеет смысла, поскольку мы не сможем проверить выполнение этого требования. Так как основной нашей задачей является использование приближений для отбора признаков, мы предъявим следующее требование: *должна иметься положительная монотонная зависимость между приближениями комбинаторных оценок и ошибками метода обучения на контроле.*

4.6.2 Описание метода

Предлагаемый метод основывается на следующей гипотезе: если взять алгоритм из нижних слоев графа расслоения-связности (например, являющийся результатом работы некоего метода обучения линейного классификатора), найти небольшое число алгоритмов, близких к нему в графе, и вычислить оценку вероятности переобучения только по этой выборке (так, как будто это все семейство), то мы получим величину, коррелирующую с ошибкой метода обучения на контроле. Далее будут приведены эксперименты, подтверждающие эту гипотезу. В данном же разделе мы опишем подход к получению такой выборки.

[9] Отметим, что приведенные доводы о коррелированности верны и для более общей задачи *выбора модели.*

В качестве начального алгоритма выберем линейный классификатор, обученный логистической регрессией, поскольку логистическая регрессия оптимизирует функционал, являющийся гладкой аппроксимацией функционала ПМЭР, и поэтому можно ожидать, что ей будет найден алгоритм, близкий к лучшему в семействе. Примем во внимание следующие наблюдения:

1. Алгоритмы с существенным вкладом в оценку концентрируются вокруг лучшего. В то же время одно случайное блуждание достаточно быстро выходит из окрестности стартовой вершины, и с крайне малой вероятностью возвращается в нее. Отсюда можно сделать вывод, что лучше делать несколько коротких блужданий, чтобы тщательнее исследовать окрестность лучшего алгоритма.

2. В нижних слоях верхняя связность вершин значительно превосходит нижнюю, поэтому случайное блуждание гораздо лучше обходит слои, лежащие выше стартового алгоритма, чем слои ниже него. Так, рассмотрим простое случайное блуждание. На рис. 2 изображен его результат; видно, что такое блуждание быстро уходит из окрестности стартового алгоритма и никогда в нее не возвращается. Более того, оно достаточно быстро уходит в верхние слои. Если же повысить вероятность смещения алгоритма вниз (рис. 3), то окрестность стартового алгоритма оказывается гораздо более исследованной.

Алгоритм 4.2 учитывает данные замечания. Отметим, что в нем на каждом шаге генерируется случайный луч, исходящий из текущей точки, и находится сосед текущего алгоритма вдоль этого луча. Это позволяет избежать трудоемких вычислений, связанных с поиском всех соседей текущего алгоритма, но в то же время сильно затрудняет теоретический анализ блуждания, поскольку вероятность попадания в вершину начинает зависеть от многих факторов.

Сложность поиска соседа вдоль луча составляет $O(Ld)$, поэтому время работы предложенного метода имеет порядок $O(Ldn)$, что позволяет его использовать для больших выборок. Отметим также, что в нем нигде не требуется общее положение выборки.

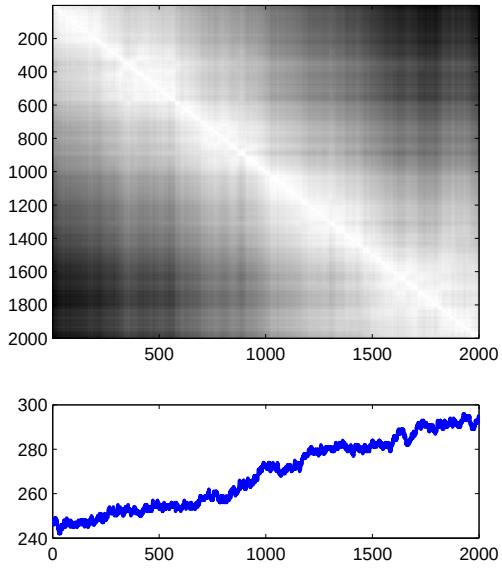


Рис. 2: Хэммингова расстояния между векторами ошибок (верхний график) и число ошибок (нижний график) алгоритмов, полученных простым случайным блужданием.

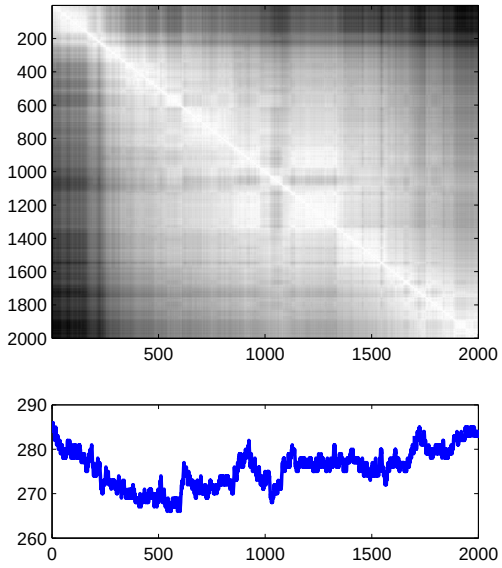


Рис. 3: Хэммингова расстояния между векторами ошибок (верхний график) и число ошибок (нижний график) алгоритмов, полученных случайным блужданием с вероятностью перехода вверх, равной 0.5.

5 Композиции линейных классификаторов

5.1 Постановка задачи

Данный раздел посвящен применению комбинаторной теории переобучения для построения композиций линейных классификаторов. Нас будет интересовать простое голосование линейных классификаторов:

$$a(x) = \operatorname{sign} \sum_{i=1}^{p} \operatorname{th}\langle w_i, x\rangle. \tag{14}$$

При этом базовые классификаторы мы будет строить над подпространствами малой размерности. Это требование, во-первых, уменьшает переобученность базовых классификаторов, что должно привести и к меньшей переобученности всей композиции, и, во-вторых, позволяет эффективно вычислять комбинаторные оценки.

Algorithm 4.2 Сэмплирование вершин

Вход: Стартовая вершина v_0, размер выборки n, число блужданий N_{RW}, вероятность перехода вверх p_{up}.

Выход: Выборка $v_1, v_2, \ldots, v_n$.

$u_i = v_0$, $i = 1, \ldots, N_{\text{RW}}$; // *текущие вершины в каждом из блужданий*

$k = 0$; // *текущие размер выборки*

цикл

  для $i = 1, \ldots, N_{\text{RW}}$

    Найти соседа u'_i алгоритма u_i вдоль случайного направления (см. раздел 3.3);

    если u'_i расположен выше u_i в графе **то**

      с вероятностью $p_{\mathbf{up}}$

        $u_i = u'_i$;

      иначе

        $u_i = u_i$;

    $k = k + 1$; $v_k = u_i$;

    если $k = n$ **то**

      выход

5.2 Использование комбинаторных оценок для отбора признаков

Пусть известна некоторая оценка вероятности переобучения

$$\mathsf{P}[\nu(\mu X, \bar{X}) - \nu(\mu X, X) \geqslant \varepsilon] \leqslant \eta(\varepsilon).$$

Обратив ее, можно оценить частоту ошибок на контроле: с вероятностью не менее $1 - \eta$

$$\nu(\mu X, \bar{X}) \leqslant \nu(\mu X, X) + \varepsilon(\eta).$$

Величину в правой части неравенства можно использовать в качестве критерия при отборе признаков или при выборе модели. Мы будем использовать ее для отбора признаков для базовых классификаторов.

Algorithm 5.1 ComBoost

Вход: Выборка X; параметры T, ℓ_0, ℓ_1;

Выход: Базовые линейные классификаторы $b_1, \ldots, b_T$

1: Инициализировать веса и отступы:
$w_i = 1$, $M_i = 0$ для всех $i = 1, \ldots, \ell$;
2: **для** $t = 1, \ldots, T$, пока не выполнен критерий останова
3: Обучить базовый алгоритм (по отобранным признакам):
$b_t := \operatorname{argmin}_b Q(b, X, W)$;
4: Обновить значения отступов:
$M_i = M_i + y_i b_t(x_i)$ для всех $i = 1, \ldots, L$;
5: упорядочить выборку X по возрастанию отступов M_i;
6: Отобрать объекты для обучения следующего базового алгоритма:
$w_i = [\ell_0 \leqslant i \leqslant \ell_1]$;

5.3 Комитетный бустинг

Композицию линейных классификаторов мы будем строить методом ComBoost (Committee Boosting, комитетный бустинг) [3]; см. алгоритм 5.1. В нем каждый из p базовых линейных классификаторов обучается по подвыборке объектов, отобранных следующим образом: в подвыборку не включаются объекты, слишком хорошо и слишком плохо классифицируемые композицией предыдущих базовых классификаторов. Отбор признаков для каждого базового классификатора будем производить путём жадного наращивания. Каждый набор признаков оценивается может оцениваться различными способами, которые будут описаны ниже.

Отметим, что большинство методов построения композиций обучают базовые классификаторы по функционалу, представляющему собой взвешенную сумму ошибок на объектах. Это затрудняет применение комбинаторных оценок, поскольку на данный момент в комбинаторной теории есть лишь техники для работы с невзвешенными суммами ошибок. По этой причине метод ComBoost отлично подходит нам, так как использует лишь единичные и нулевые веса, что соответствует исключению определенных объектов из выборки.

5.4 Моделирование логистической регрессии с помощью метода минимизации эмпирического риска

Эксперименты, представленные в данном разделе, преследуют две цели. Во-первых, мы выясним, насколько хорошо комбинаторный функционал вероятности переобучения $Q_\varepsilon(\mu, \mathbb{X})$, вычисленный с помощью случайных блужданий, позволяет оценить обобщающую способность логистической регрессии. Во-вторых, мы сравним комбинаторные оценки с непосредственно вычисленным по Монте-Карло функционалом $Q_\varepsilon(\mu, \mathbb{X})$, а также с современными PAC-Bayes оценками [13].

Таблица 1: Описание задач из репозитория UCI.

Задача	L	d	Задача	L	d
Sonar	208	60	Glass	214	9
Liver dis.	345	6	Ionosphere	351	34
Wdbc	569	30	Australian	690	6
Pima	768	8	Faults	1941	27
Statlog	2310	19	Wine	4898	11
Waveform	5000	21	Pageblocks	5473	10
Optdigits	5620	64	Pendigits	10992	16
Letter	20000	16			

Эксперименты будем проводить на 15 задачах из репозитория UCI, описание которых дано в таблице 1.

В каждом эксперименте будем разбивать генеральную выборку $\mathbb{X}$ на обучающую $\mathbb{X}_L$ и контрольную $\mathbb{X}_K$. Обучающую выборку $\mathbb{X}_L$ будем использовать для обучения логистической регрессии и для вычисления комбинаторных оценок. Затем будем сравнивать истинную ошибку логистической регрессии на скрытой выборке $\mathbb{X}_K$ с полученными оценками.

Введем следующие обозначения для средней по всем разбиениям ошибки на обучении и контроле:

$$\nu_\ell(\mu, \mathbb{X}) = \frac{1}{C_L^\ell} \sum_{\mathbb{X}=X_\ell \sqcup X_k} \nu(\mu X, X); \tag{15}$$

$$\bar{\nu}_\ell(\mu, \mathbb{X}) = \frac{1}{C_L^\ell} \sum_{\mathbb{X}=X_\ell \sqcup X_k} \nu(\mu X, \bar{X}). \tag{16}$$

В первом эксперименте мы получим кривые обучения логистической регрессии, варьируя длину обучающей выборки L от 5% до 95% от размера генеральной выборки. Для каждого L сгенерируем $M = 100$ разбиений $\mathbb{X} = \mathbb{X}_L^i \sqcup \mathbb{X}_K^i,\ i = 1, \ldots, M$ и вычислим по ним среднюю частоту ошибок логистической регрессии μ_{LR} на обучении $\nu_L(\mu_{\text{LR}}, \mathbb{X})$ и контроле $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$ методом Монте-Карло:

$$\hat{\nu}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M} \sum_{i=1}^{M} \nu(\mu_{\text{LR}} \mathbb{X}_L^i, \mathbb{X}_L^i), \qquad \hat{\bar{\nu}}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M} \sum_{i=1}^{M} \nu(\mu_{\text{LR}} \mathbb{X}_L^i, \mathbb{X}_K^i).$$

Далее, для каждой обучающей выборки $\mathbb{X}_L$ мы получим с помощью случайного блуждания выборку алгоритмов и оценим по ней частоты ошибок метода ПМЭР μ $\nu_\ell(\mu, \mathbb{X}_L)$ и $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$. Для сэмплирования алгоритмов воспользуемся алгоритмом 4.2 с параметрами $n = 8\,192$, $N_{\text{RW}} = 64$, $p_{\text{up}} = 0.8$, при этом в качестве стартовой вершины возьмем алгоритм $\mu_{\text{LR}} \mathbb{X}_L$. Затем вычислим оценки на величины $\nu_\ell(\mu, \mathbb{X}_L)$ and $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$ методом Монте-Карло по $M' = 4\,096$ разбиений $\mathbb{X}_L = X_\ell^j \sqcup X_k^j$, $j = 1, \ldots, M'$, при этом возьмем такое ℓ, что $\frac{\ell}{L} = 0.8$:

$$\hat{\nu}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M} \nu(\mu X_\ell^j, X_\ell^j), \qquad \hat{\bar{\nu}}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M} \nu(\mu X_\ell^j, X_k^j).$$

Данные оценки усредним по всем разбиениям $\mathbb{X} = \mathbb{X}_L^i \sqcup \mathbb{X}_K^i$.

Итоговые четыре величины (истинные ошибки логистической регрессии на обучении и контроле $\nu_L(\mu_{\text{LR}}, \mathbb{X})$ and $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$, ошибка ПМЭР на обучении $\nu_\ell(\mu, \mathbb{X}_L)$ и ошибка ПМЭР на контроле $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$) на рис. 4 как функции от размера обучающей выборки. Видно, что ошибка ПМЭР $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$, вычисленная лишь по обучающей выборке $\mathbb{X}_L$, дает хорошую оценку на истинную ошибку логистической регрессии $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$ на скрытой выборке $\mathbb{X}_K$. Отметим также, что кривые обучения ПМЭР хорошо восстанавливают форму кривых обучения логистической регрессии.

Перейдем теперь к сравнению различных оценок обобщающей способности. Для каждой задачи будем усреднять результаты по 100 разбиениям на обучение $\mathbb{X}_L$ и

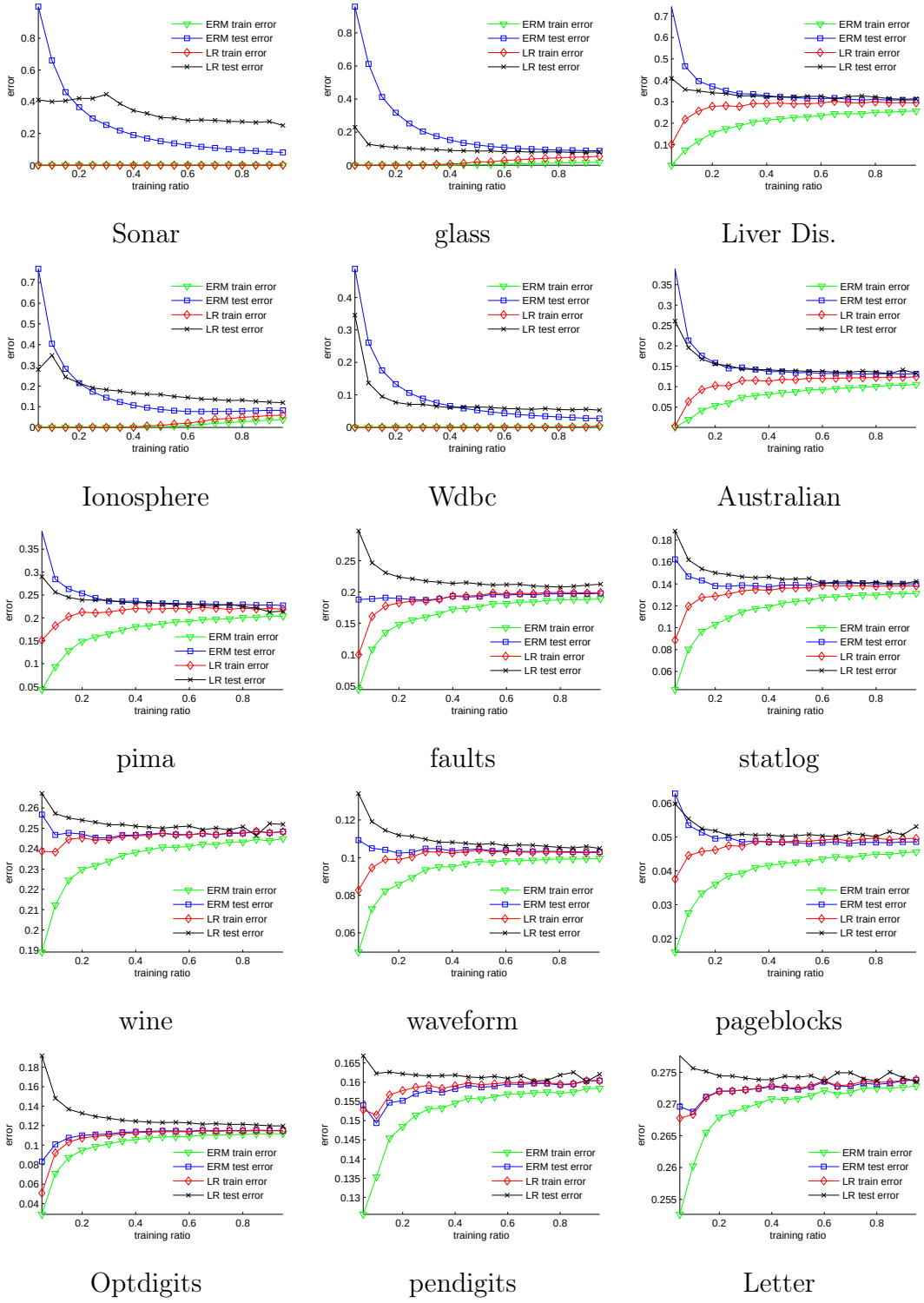


Рис. 4: Кривые обучения логистической регрессии и ПМЭР. Частота ошибок логистической регресси оценивается по разбиениям генеральной выборки $\mathbb{X} = \mathbb{X}_L \sqcup \mathbb{X}_K$. Частота ошибок ПМЭР оценивается по разбиениям обучающей выборки $\mathbb{X}_L = X_\ell \sqcup X_k$.

контроль $\mathbb{X}_K$. Как и в предыдущем эксперименте, мы используем обучающую вы-

	Оценки Монте-Карло				Оценки обобщающей способности			
Task	TrainErr	TestErr	Overfit	$\delta_\ell(\mu)$	VC	SC	PAC DI	PAC DD
Sonar	0.000	0.271	0.271	0.095	0.185	0.119	1.287	1.287
glass	0.046	0.075	0.029	0.078	0.211	0.140	1.126	0.738
Liver dis.	0.299	0.314	0.015	0.060	0.261	0.209	1.207	1.067
Ionosphere	0.049	0.125	0.077	0.052	0.150	0.112	1.219	1.153
Wdbc	0.001	0.056	0.055	0.032	0.071	0.043	1.174	0.705
Australian	0.122	0.136	0.013	0.030	0.137	0.110	1.146	0.678
pima	0.220	0.227	0.007	0.028	0.159	0.127	0.971	0.749
faults	0.198	0.210	0.012	0.010	0.108	0.087	1.110	1.061
statlog	0.138	0.142	0.005	0.010	0.096	0.082	1.102	0.747
wine	0.248	0.250	0.002	0.004	0.134	0.109	0.776	0.637
waveform	0.103	0.105	0.002	0.004	0.099	0.079	0.561	0.354
pageblocks	0.050	0.050	0.001	0.004	0.073	0.057	0.737	0.186
Optdigits	0.115	0.121	0.006	0.004	0.102	0.084	1.068	0.604
pendigits	0.160	0.161	0.001	0.002	0.127	0.103	0.774	0.432
Letter	0.274	0.274	0.001	0.001	0.165	0.137	0.818	0.636

Таблица 2: Сравнение истинной переобученности логистической регрессии с различными оценками. Через TrainErr обозначена величина $\nu_L(\mu_{\text{LR}}, \mathbb{X})$, через TestErr — $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$, Overfit равно их разности, $\delta_\ell(\mu) \equiv \bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$. Столбец SC соответствует оценке расслоения-связности.

борку $\mathbb{X}_L$ для обучения логистической регрессии, сэмплирования алгоритмов методом 4.2 и оценивания величин $\nu_\ell(\mu, \mathbb{X}_L)$ и $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$ по 4 096 разбиениям $\mathbb{X}_L = X_\ell^j \sqcup X_k^j$. Также будем оценивать по выборке $\mathbb{X}_L$ переобученность $\bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$ с помощью медиан VC-оценки (1) и оценки расслоения-связности (2). Наконец, будем вычислять по обучающей выборке PAC-Bayes DI- и DD-оценки, предложенные в [13]. Результаты представлены в таблице 2. Отметим, что все комбинаторные оценки приближают переобучение, тогда как PAC DI и PAC DD являются верхними оценками на частоту ошибок на контроле.

Видно, что оценка $\delta_\ell(\mu) \equiv \bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$ является на порядки более точной по сравнению со всеми остальными, и хорошо оценивает переобучение на всех зада-

чах, кроме Sonar (отметим, что она является самой маленькой среди всех). Оценка расслоения-связности значительно точнее VC-оценки, но все равно является весьма завышенной. Все комбинаторные оценки являются более точными по сравнению с PAC-Bayes оценками.

5.5 Эксперименты с построением композиций [10]

В качестве базовых алгоритмов будем использовать линейные классификаторы, настроенные методом SVM. Каждый базовый классификатор обучается над подпространством размерности не больше 5, которое строится путем жадного наращивания, где в качестве оценки качества подпространства выступает один из критериев, описанных ниже.

1. **Обращенные комбинаторные оценки.** Сначала производится сэмплирование вершин с помощью подхода на основе метода Frontier Sampling, подробно описанного в разделе 4.5. Затем по формуле (11) приближенно вычисляется либо оценка (2), либо оценка (4), которые затем обращаются при $\eta = \frac{1}{2}$ методом дихотомии. Наконец, вычисляется комбинаторный критерий по формуле

$$Q_c = \nu(a_0, X) + \varepsilon\left(\tfrac{1}{2}\right), \tag{17}$$

   где a_0 — лучший из найденных классификаторов. Данный критерий будем обозначать либо как CombQepsSc, либо как CombQepsScSources (для оценок (2) и (4) соответственно).

2. **Оценка скользящего контроля.** По обучающей выборке вычисляется оценка скользящего контроля для метода SVM по 100 разбиениям на две равные части. Будем обозначать данный критерий как CV.

3. **Эмпирический риск.** В качестве критерия выступает эмпирический риск $\nu(a, \mathbb{X})$ алгоритма a, построенного методом SVM. Будем обозначать данный критерий как Emp.

[10] Результаты, описанные в данном разделе, были представлены на конференции ИОИ-2012

	Wine Quality	Statlog	Waveform	Faults
NN	72,06	85,41	86,79	74,51
Emp	64,70	84,92	84,78	73,39
CV	71,06	85,26	86,38	75,76
CombQepsSc	69,48	86,26	85,77	**77,81**
CombQepsScSources	**74,68**	**86,75**	**86,91**	74,03

Таблица 3: Результаты экспериментов с построением композиций. Показан процент правильных ответов на контрольной выборке. **Жирным** выделен лучший результат на каждой задаче.

Также представляет интерес сравнение с двухслойной нейронной сетью, поскольку структурно она совпадает с композициями, настраиваемыми предложенным методом. А именно, в терминах нейронных сетей предложенный подход строит сильно разреженную двухслойную нейронную сеть с автоматическим выбором числа нейронов в скрытом слое. Мы будем пользоваться методом обратного распространения для настройки нейросети. Данный метод будем обозначать через NN.

Для проведения экспериментов было выбрано несколько наборов данных из репозитория UCI. Для каждого набора в обучающую выборку отбирается 250 объектов, остальные составляли тестовую выборку. В ComBoost использовались фиксированные параметры $\ell_0 = 5$, $\ell_1 = 150$. Длина композиции была ограничивалась 15-ю классификаторами. Результаты представлены в таблице 3.

Из представленных результатов можно сделать следующие выводы:

1. Использование критериев, предсказывающих ошибку на контроле, дает выигрыш по сравнению с подходом, использующим лишь эмпирический риск (CV, CombQepsSc и CombQepsScSources по сравнению Emp).

2. Использование комбинаторных оценок дает выигрыш по сравнению с оценками скользящего контроля (CV по сравнению с CombQepsSc и CombQepsScSources).

3. Оценка (4), учитывающая связи между несравнимыми алгоритмами, дает улучшения по сравнению с оценкой расслоения-связности (2).

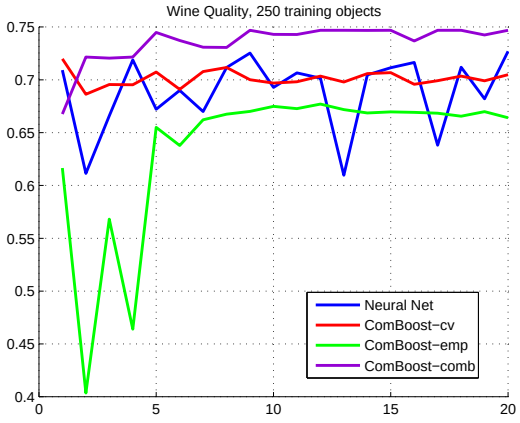


Рис. 5: Зависимость доли правильных ответов на контроле от длины композиции. Задача Wine.

4. Предложенный подход для построения композиций с помощью ComBoost дает выигрыш по сравнению с настройкой двухслойных нейросетей методом обратного распространения ошибки.

Также на рис. 5 изображена зависимость качества композиции от длины выборки. Видно, что при использовании комбинаторных критериев удается достичь хорошего качества классификации уже при небольшом числе базовых алгоритмов.

Таким образом, тщательный контроль переобучения при отборе признаков в базовых классификаторах позволяет, в отличие от стандартных методов типа бустинга и бэггинга, обходиться малым числом базовых классификаторов.

5.6 Эксперименты с построением композиций-2

Метод вычисления комбинаторных оценок, использованный в предыдущем разделе, обладает рядом существенных недостатков. Во-первых, он работает крайне медленно, и для построения композиции на одной задаче требуется порядка 24 часов. Во-вторых, он использует метод поиска соседей линейного классификатора, требующий, чтобы выборка находилась в общем положении. В разделе 4.6 был предложен метод вычисления оценок, лишенный этих недостатков; в тоже время не существует каких-либо гарантий корректности этого метода. Данный раздел посвящен его экспериментальному изучению.

5.6.1 Критерии отбора признаков

Перечислим критерии отбора признаков, которые мы будем использовать в данном эксперименте.

1. С помощью алгоритма 4.2 с параметрами $n = 4096$, $N_{\text{RW}} = 256$ сэмплируется выборка алгоритмов. Далее либо по ней вычисляется оценка (4), которая затем обращается при $\eta = \frac{1}{2}$ (критерий CombQeps), либо методом монте-карло оценивается ошибка на контроле сэмплированного семейства (критерия CombCV).

2. Ошибка на контроле логистической регрессии, вычисленная по монте-карло (критерий CV).

3. Эмпирический риск классификатора, построенного логистической регрессией (критерий Emp).

4. Критерий, основанный на PAC-Bayes оценке (PAC-Bayes DD). Описание критерия будет дано ниже.

5.6.2 Связь между оценками и ошибкой на контроле

Для того, чтобы оценки, получаемые в результате сэмплирования, можно было использовать для отбора признаков, необходимо, чтобы имелась монотонная зависимость между оценками и ошибкой на контрольной выборке. Чтобы проверить выполнение этого требования, возьмем задачу `statlog`, выберем из нее 250 объектов, отложив остальные в контроль, и запустим жадный отбор признаков. На каждом шаге этого отбора рассматривается несколько подмножеств-кандидатов, для которых вычисляется критерий CombCV. Мы используем оценку монте-карло вместо комбинаторных оценок, чтобы исключить влияние завышенности оценок на результат.

На рис. 6 изображены результаты экспериментов. Видно, что критерию CombCV удается с некоторой точностью воспроизвести ранжирование, задаваемое частотой ошибок на контроле.

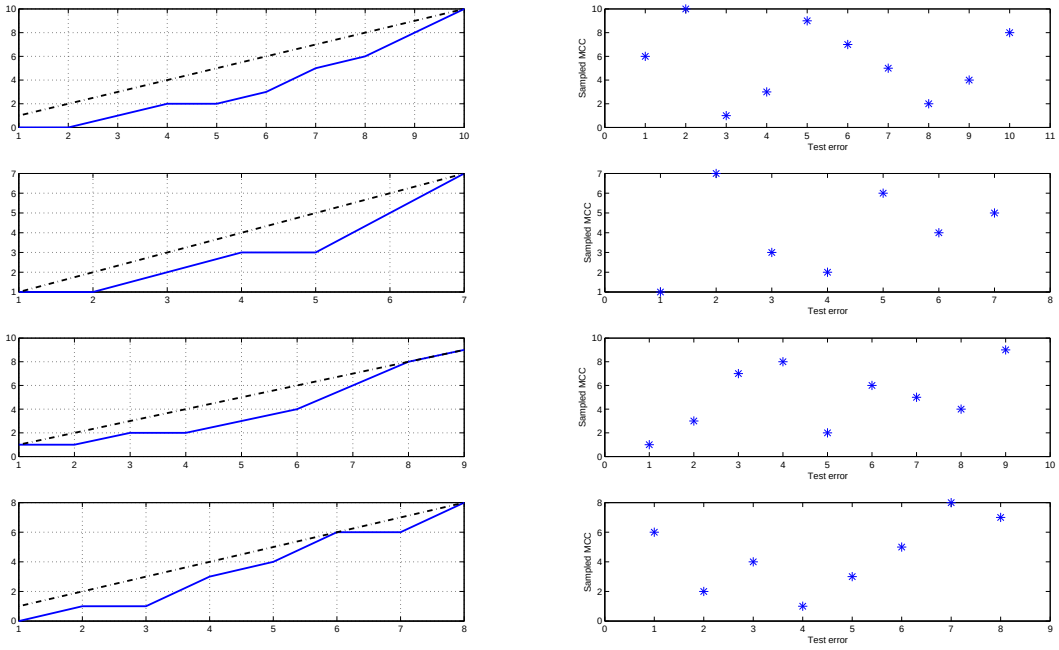


Рис. 6: Результаты сравнения оценок и ошибок на контроле. Справа: по оси X отложены ранги признаковых подпространств согласно комбинаторным оценкам, по оси Y — ранги тех же подпространств согласно ошибке логистической регрессии на контроле. Слева: значение графика в точке k — это число подпространств среди первых k в ранжировании по комбинаторному критерию, которые также попали в первые k среди лучших с точки зрения ошибки на контроле. В идеальном случае эта кривая должна иметь вид $y = x$.

5.6.3 PAC-Bayes оценки

Одной из лучших на данный момент оценок обобщающей способности для линейных классификаторов является PAC-Bayes оценка, учитывающая размерность задачи [13]. Она имеет вид

$$\mathrm{kl}(2\nu(a,\bar{X}) \parallel f(w,X,Y)) \leqslant \frac{\frac{d}{2}\ln\left(1+\frac{\mu^2}{d}\right)+\ln\frac{\ell+1}{\delta}}{\ell}, \tag{18}$$

где $\mathrm{kl}(a\|b) = a\ln\frac{a}{b} + (1-a)\ln\frac{1-a}{1-b}$, a — линейный классификатор с вектором весов w, $f(w,X,Y) = \frac{1}{\ell}\sum_{i=1}^{\ell}\bar{\Phi}(\mu\gamma(w,x_i,y_i))$, $\gamma(w,x,y) = y\frac{\langle w,x\rangle}{\|x\|}$, $\bar{\Phi}$ — вероятность правого хвоста нормального распределения, μ — произвольное положительное число.

Решая неравенство (18) относительно $\nu(a,\bar{X})$ и оптимизируя его по μ, можно получить оценку на частоту ошибок классификатора a на скрытой выборке.

	statlog	waveform	wine	faults
CombCV	**85,35**	87,56	**71,63**	**73,62**
CombQeps	**85,08**	**87,66**	71,08	**71,65**
CV	84,04	**88,13**	**71,52**	70,86
Emp	80,77	87,34	71,49	71,09
PacBayes DD	82,13	87,17	64,68	67,67

Таблица 4: Результаты экспериментов с построением композиций. Показан процент правильных ответов на контрольной выборке (с усреднением по 5 разбиениям). **Жирным** выделены два лучших результата на каждой задаче.

5.6.4 Построение композиций

Как и в предыдущем эксперименте, для построения композиций будем использовать ComBoost, но на этот раз базовые классификаторы будем настраивать с помощью логистической регрессии. Для каждого набора в обучающую выборку будем отбирать 250 объектов, остальные будут составлять тестовую выборку. В ComBoost возьмем фиксированные параметры $\ell_0 = 5$, $\ell_1 = 150$. Длину композиции ограничим 10-ю классификаторами. Результаты экспериментов будем усреднять по 5 разбиениям генеральной выборки на обучение и контроль. Результаты представлены в таблице 4.

Подход на основе сэмплирования всегда дает один из двух лучших результатов на каждой задаче.

6 Заключение

В рамках данной работы были получены следующие результаты:

1. Предложена новая комбинаторная оценка вероятности переобучения, учитывающая попарные взаимодействия алгоритмов.

2. Предложен метод для обхода графа расслоения-связности линейных классификаторов.

3. Предложен метод эффективного вычисления комбинаторных оценок, основанный на случайных блужданиях.

4. Показано, что комбинаторные оценки для функционала минимизации эмпирического риска, вычисленные для семейства линейных классификаторов, тесно связаны с обобщающей способностью логистической регрессии.

5. Предложенные методы использованы для построения композиций линейных классификаторов, показано их преимущество по сравнению с современными оценками вероятности переобучения.

7 Список публикаций

1. Riabenko E., Kogadeeva M., Gavrilyuk K., Sokolov E., Shanin I., Tonevitsky A. Comparing Affymetrix Human Gene 1.0 ST preprocessing methods on tissue mixture data. // *6th International Conference on Bioinformatics and Biomedical Engineering (iCBBE), 2012. Pp. 631–634. Shanghai, China.*

2. Соколов Е. А. Оценки вероятности переобучения и комбинаторные отступы в задачах классификации. // *Сборник тезисов XIX Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов-2012». Секция «Вычислительная математика и кибернетика». М.: МАКС Пресс, 2012, с.106-107.*

3. Соколов Е. А., Воронцов К. В. Минимизация вероятности переобучения для композиций линейных классификаторов малой размерности. // *Сборник докладов 9-й международной конференции «Интеллектуализация обработки информации». М.: Торус Пресс, 2012, с. 82-85.*

Список литературы

[1] *Ботов, П.* Точные оценки вероятности переобучения для монотонных и унимодальных семейств алгоритмов / П. Ботов // 14-я Всеросс. конф. Математические методы распознавания образов. — М.: МАКС Пресс, 2009. — Pp. 7–10.

[2] *Ботов, П.* Уменьшение вероятности переобучения итерационных методов статистического обучения / П. Ботов // 15-я Всеросс. конф. Математические методы распознавания образов. — М.: МАКС Пресс, 2011. — Pp. 44–47.

[3] *Маценов, А.* Комитетный бустинг: минимизация числа базовых алгоритмов при простом голосовании / А. Маценов // Всероссийская конференция ММРО-13. — М.: МАКС Пресс, 2007. — Pp. 180–183.

[4] *Толстихин, И.* Вероятность переобучения плотных и разреженных семейств алгоритмов / И. Толстихин // 8-я Межд. конф. Интеллектуализация обработки информации. — М.: МАКС Пресс, 2010. — Pp. 83–86.

[5] *Avrachenkov, K.* Improving random walk estimation accuracy with uniform restarts / K. Avrachenkov, B. Ribeiro // Proc. of WAW. — 2010.

[6] *Balinski, M.* On the graph structure of convex polyhedra in n-space / M. Balinski // *Pac. J. Math.* — 1961. — Vol. 11. — Pp. 431–434.

[7] *Bartlett, M. S.* An Inverse Matrix Adjustment Arising in Discriminant Analysis / M. S. Bartlett // *The Annals of Mathematical Statistics.* — 1951. — March. — Vol. 22, no. 1. — Pp. 107–111.

[8] *Botov, P.* Exact estimates of the probability of overfitting for multidimensional modeling families of algorithms / P. Botov // *Pattern Recognition and Image Analysis.* — 2011. — Vol. 21, no. 1. — Pp. 52–65.

[9] *Diaconis, P.* Geometric Bounds for Eigenvalues of Markov Chains / P. Diaconis, D. Stroock // *The Annals of Applied Probability.* — 1991. — February. — Vol. 1, no. 1. — Pp. 36–61.

[10] *Frei, A.* Accurate estimates of the generalization ability for symmetric set of predictors and randomized learning algorithms / A. Frei // *Pattern Recognition and Image Analysis.* — 2010. — Vol. 20, no. 3. — Pp. 241–250.

[11] *Grimmett, G.* Probability and random processes / G. Grimmett, D. Stirzaker. — 2001.

[12] *Grunbaum, B.* Convex polytopes / B. Grunbaum. — Springer, 2003.

[13] *Jin, C.* Dimensionality Dependent PAC-Bayes Margin Bound / C. Jin, L. Wang // *Advances in Neural Information Processing Systems.* — 2012. — Vol. 25. — Pp. 1043–1051.

[14] *Koltchinskii, V.* Local Rademacher complexities and oracle inequalities in risk minimization (with discussion) / V. Koltchinskii // *The Annals of Statistics.* — 2006. — Vol. 34. — Pp. 2593–2706.

[15] *Koltchinskii, V.* Empirical margin distributions and bounding the generalization error of combined classifiers / V. Koltchinskii, D. Panchenko // *The Annals of Statistics.* — 2002. — Vol. 31, no. 1. — Pp. 1–50.

[16] *Koltchinskii, V.* Bounding the generalization error of convex combinations of classifiers: balancing the dimensionality and the margins / V. Koltchinskii, D. Panchenko // *The Annals of Applied Probability.* — 2003. — Vol. 13, no. 1. — Pp. 213–252.

[17] *Lee, C.-h.* Beyond Random Walk and Metropolis-Hastings Samplers: Why You Should Not Backtrack for Unbiased Graph Sampling / C.-h. Lee, X. Xu, D. Y. Eun. — 2012. — Pp. 1–36.

[18] *Leopardi, P.* A partition of the unit sphere into regions of equal area and small diameter / P. Leopardi // *Electronic Transactions on Numerical Analysis.* — 2006. — Vol. 25. — Pp. 309–327.

[19] *Lugosi, G.* Concentration-of-Measure Inequalities: Tech. rep. / G. Lugosi: Machine Learning Summer School, Australian National University, Canberra, 2004.

[20] *Ribeiro, B.* Estimating and sampling graphs with multidimensional random walks / B. Ribeiro, D. Towsley // *Proceedings of the 10th annual conference on Internet measurement - IMC '10.* — 2010. — P. 390.

[21] *Roberts, G. O.* General state space Markov chains and MCMC algorithms / G. O. Roberts, J. S. Rosenthal // *Probability Surveys.* — 2004. — Vol. 1. — Pp. 20–71.

[22] *Sinclair, A.* Improved bounds for mixing rates of Markov chains and multicommodity flow / A. Sinclair. — 1992. — Vol. 583, no. April.

[23] *Vapnik, V.* On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities / V. Vapnik, A. Chervonenkis // *Theory of Probability and its Applications*. — 1971. — Vol. 16, no. 2. — Pp. 264–280.

[24] *Vorontsov, K.* Exact combinatorial bounds on the probability of overfitting for empirical risk minimization / K. Vorontsov // *Pattern Recognition and Image Analysis*. — 2010. — Vol. 20, no. 3. — Pp. 269–285.

[25] *Vorontsov, K.* Tight combinatorial generalization bounds for threshold conjunction rules / K. Vorontsov, A. Ivahnenko // 4-th Int'l Conf. on Pattern Recognition and Machine Intelligence (PReMI'11). — Lecture Notes in Computer Science. Springer-Verlag, 2011. — Pp. 66–73.

[26] Walking on a Graph with a Magnifying Glass: Stratified Sampling via Weighted Random Walks / M. Kurant, M. Gjoka, C. T. Butts, A. Markopoulou. — 2011. — Vol. 1.