Случайные признаки в машинном обучении

И.Р. Садртдинов, Е.М. Шабалин

Многих алгоритмы машинного обучения так или иначе используют случайность. Это может быть, например, бутстрэп в случайных лесах или случайная инициализация весов нейросетей. В этой работе мы разберем подход, основанный на использовании рандомизированных признаков. Конечно, случайности сложно соперничать с общепризнанными гигантами среди моделей вроде градиентного бустинга или многослойных нейросетей, но оказывается, что применение рандомизированного подхода может привести к улучшению качества моделей.

1 Random Fourier Features (RFF)

Использование ядрового перехода позволяет использовать нелинейные преобразования признаков, что приводит к снижению смещения моделей. Однако, введение ядер заметно усложняет вычислительную сложность обучения и применения алгоритмов. Здесь на помощь приходит теорема Бохнера, и некоторые ядра (инвариатные относительно сдвига, shift-invariant kernels) удается представить через преобразование Фурье по определенному вероятностному распределению (будем считать, что $x, y \in \mathbb{R}^d$).

$$K(x,y) = K(x-y) = \int_{\mathbb{R}^d} e^{iw^T(x-y)} p(w) dw = \int_{\mathbb{R}^d} \cos\left(w^T(x-y)\right) p(w) dw$$

Этот интеграл можно оценить методом Монте-Карло. Если преобразовать косинус разности, можно перейти к скалярному произведению объектов в некотором другом пространстве.

$$K(x-y) = \mathbb{E}_{w\sim p}\Big[\cos\left(w^T(x-y)\right)\Big] \approx \sum_{i=1}^{D} \cos\left(w_i^T(x-y)\right) =$$

$$= \sum_{i=1}^{D} \cos\left(w_i^T x\right)\cos\left(w_i^T y\right) + \sin\left(w_i^T x\right)\sin\left(w_i^T y\right)$$

Новые признаковые описания выглядят так (тут вводится нормализация с помощью деления на $\sqrt{D}$).

$$\widetilde{x} = \frac{1}{\sqrt{D}}\Big[\cos\left(w_1^T x\right), \ldots, \cos\left(w_D^T x\right), \sin\left(w_1^T x\right), \ldots, \sin\left(w_D^T x\right)\Big]$$

Получаем аппроксимацию ядра:

$$K(x-y) \approx \langle \widetilde{x}, \widetilde{y} \rangle$$

Однако на практике лучший результат показывают признаки, сконструированные немного по-другому. Поскольку $\sin\alpha = \cos\left(\alpha - \frac{\pi}{2}\right)$, то синусы можно аппроксимировать, добавив в косинусы внутренние сдвиги b_i, распределенные равномерно на $[-\pi, \pi]$:

$$\overline{x} = \sqrt{\frac{2}{D}}\Big[\cos\left(w_1^T x + b_1\right), \ldots, \cos\left(w_D^T x + b_D\right)\Big]$$

В дальнейшем число случайных признаков будем обозначать за D. В итоге преобразование в новое признаковое пространство выглядит так:

$$\widetilde{x} = \cos\left(Wx + b\right)$$

$$W \in \mathbb{R}^{D\times d}, W_i \sim p \qquad b \in \mathbb{R}^D, b_i \sim U[-\pi, \pi]$$

1.1 Алгоритм

В статье Ali Rahimi, Benjamin Recht приводятся распределения для трех ядер.

Ядро	$K(\Delta)$	$p(w)$
Гаусс	$e^{-\frac{\|\Delta\|_2^2}{2}}$	$(2\pi)^{-\frac{d}{2}} e^{-\frac{\|w\|_2^2}{2}}$
Лаплас	$e^{-\|\Delta\|_1}$	$\prod_i \frac{1}{\pi\left(1+w_i^2\right)}$
Коши	$\prod_i \frac{2}{1+\Delta_i^2}$	$2^{-d} e^{-\|w\|_1}$

В первую очередь необходимо оценить дисперсию для этих распределений. Обозначим за $\mathbb{X}$ распределение объектов выборки. Обозначим за s среднюю норму разности между объектами выборки, оценим ее по некоторой подвыборке обучающей выборки. Параметр масштаба для всех распределений примем одинаковым и равным $\sigma = \frac{1}{s}$.

$$s = \mathbb{E}_{x,y\sim\mathbb{X}}\Big[\|x-y\|_2\Big] \approx \frac{1}{L^2}\sum_{i,j=1}^{L}\|x_i - x_j\|_2$$

Затем сэмплируем случайную матрицу $W \in \mathbb{R}^{D\times d}$ и вектор сдвигов $b \in \mathbb{R}^D$. Каждый элемент матрицы можно просемплировать из соответствующего одномерного распределения (мы рассматриваем покомпонентно независимые многомерные распределения). Далее генерируем новые признаки $\widetilde{x} = \cos(Wx + b)$ и обучаем на них логистическую регрессию.

1.2 Ортогональные признаки (ORF)

Хорошо известно, что многие алгоритмы машинного обучения работают лучше, если признаки объектов нескоррелированы. Для случайных признаков можно добиться похожего эффекта. Разберем подробно этот подход на примере ядра Гаусса, описанный в статье Felix Xinnan Yu et al.

Пусть элементы матрицы $G \in \mathbb{R}^{d \times d}$ распределены по стандартному нормальному закону, $G_{ij} \sim \mathcal{N}(0, 1)$. Обычный метод случайных признаков Фурье использует матрицу $W = \sigma G$. Попробуем использовать случайную ортогональную матрицу Q, распределенную равномерно на множестве ортогональных матриц ($Q \sim U(O(d))$). Как известно, столбцы матрицы Q образуют ортонормированный базис. Теперь можно было бы использовать матрицу $W = \sigma Q$, но такие признаки не будут аппроксимировать ядро Гаусса.

Дело в том, что нормы строк матрицы G имеют хи-распределение:

$$\|G_i\|_2 = \sqrt{\sum_{j=1}^{d} G_{ij}^2} \sim \chi(d)$$

В то же время, нормы строк матрицы Q равны 1. Однако, можно уравнять распределения норм строк, если домножить матрицу Q слева на диагональную матрицу S, распределенную по $\chi(d)$:

$$S_{ij} = \begin{cases} s_i \sim \chi(d), & i = j \\ 0, & i \neq j \end{cases}$$

Ортогональные признаки Фурье считаются по формуле $\widetilde{x} = \cos(Wx + b) = \cos(\sigma SQ + b)$.

Осталось выяснить, как генерировать случайную ортогональную матрицу Q. Утверждается, что можно взять случайную нормальную матрицу G, применить к ней QR-разложение, и получившаяся матрица Q будет распределена равномерно на множестве ортогональных матриц.

Стоит отметить, что на практике $D \leq d$ признаков недостаточно для получения хорошего качества, однако процедуру, описанную выше, можно повторить несколько раз, пока не будет получено желаемое число признаков.

1.3 Регуляризация

Также мы вывели эмпирическую закономерность, которую можно использовать для регуляризации модели. Ранее мы вводили масштабирующий коэффициент как $\sigma = \frac{1}{s}$. Можно фиксировать некоторый параметр λ и ввести параметр масштаба как $\sigma = \frac{\lambda}{s}$. С уменьшением разброса случайных признаков модели становится сложнее подгоняться под обучающую выборку. Этот эффект подтверждается экспериментами.

2 Данные

За основу мы взяли данные о классификации картин импрессионистов с kaggle.com. Данные содержат картины десяти художников, заранее разделенные на обучающую и валидационную выборку. При работе это разделение было сохранено. Для оценки качества мы использовали метрики accuracy и top-3 accuracy. Примеры картин из обучающей выборки представлены на рис. 1.

Рис. 1: Экземпляры картин из обучающей выборки (как вы уже поняли, слева - Клод Моне, справа - Ван Гог).

3 Convolutional encoder

Сложно поспорить с тем, что сверточные нейросети очень хороши в извлечении признаков из картинок. Для достижения лучшего качества мы обучили небольшую нейросеть, которую использовали как энкодер эмбеддингов для исходных картинок.

3.1 Архитектура

Мы использовали четыре сверточных слоя с возрастающим числом каналов (32, 64, 128, 256) и фильтрами 3x3 с последующими слоями субдискретизации (максимум, фильтр 2x2). Выход последнего сверточного слоя подвергается поканальному усреднению (global average pooling). Далее следуют 2 полносвязных слоя на 128 единиц, а всю конструкцию венчает полносвязный слой с Softmax на 10 классов. В качестве промежуточной активации использовалась LeakyRelu с параметром $\alpha = 0.1$. Подробно архитектуру можно увидеть на рис. 2.

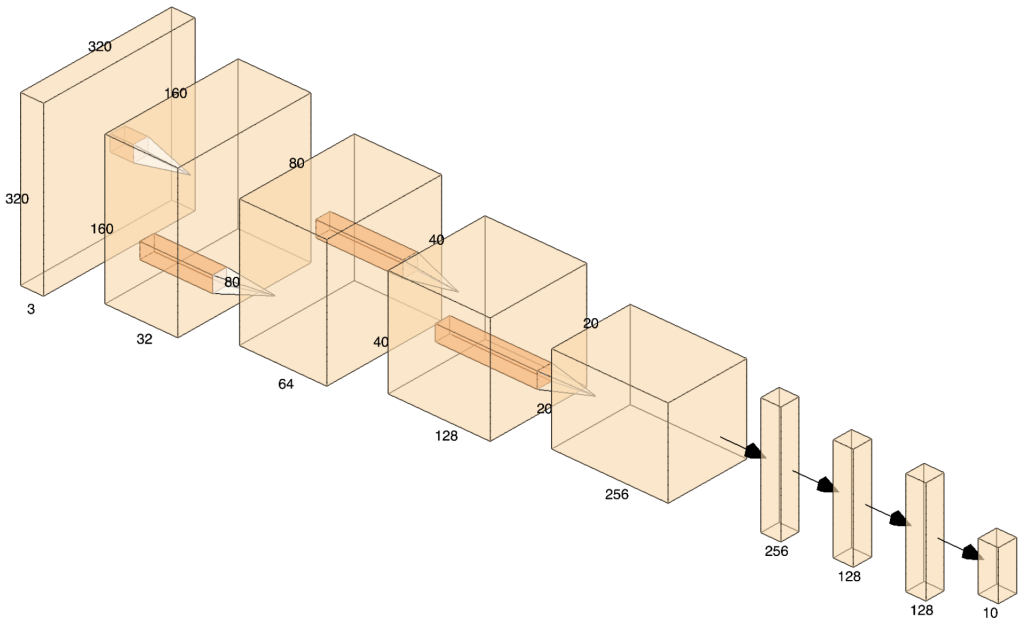

Рис. 2: Архитектура сверточного энкодера.

3.2 Обучение

Для подачи на вход сети картинки подгонялись под размер 320x320, в качестве функции потерь выступала категориальная кросс-энтропия. Для регуляризации сети использовался дропаут. Поскольку данных для задачи немного, аугментация хотя бы немного помогает от переобучения сети. Мы использовали нормальный шум, дисперсия которого увеличивалась на более поздних эпохах обучения. Еще один метод аугментации мы позаимствовали у Nitin Viswanathan. Заключается он в том, что при обучении картинки случайно переворачиваются по горизонтали. В конце концов, сеть все равно получилась переобученной, но разрыв между качеством на обучающей и валидационной выборке оказался не столь разительным:

	accuracy	top-3 accuracy
train	0.644	0.891
test	0.592	0.848

3.3 Эмбеддинги

В качестве эмбеддингов для картинок мы использовали выход последнего сверточного слоя (после поканального усреднения). Таким образом, получились вектора размерности 256. Их мы и использовали как основу для генерации случайных признаков.

4 Эксперименты

4.1 Число признаков

Мы провели серию экспериментов, в которой проверяли прирост качества от увеличения числа случайных признаков для разных ядер. Результаты убеждают в том, что модель имеет очень низкое смещение: она способна подгоняться под любые данные, что выражается в практически идеальном качестве на обучающей выборке.

Хуже всего себя показало ядро Лапласа (скорее всего это связано со свойствами распределения Коши, которое используется для аппроксимациии, в частности, у него отсутствует матожидание и дисперсия). Ядра Коши и Гаусса показали куда лучший результат с заметным превосходством последнего. В целом, качество ожидаемо растет с увеличением числа случайных признаков.

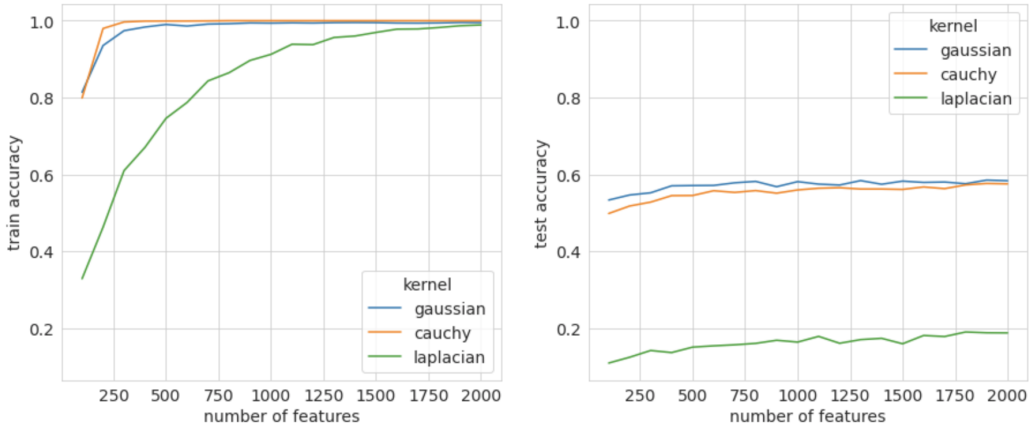

Рис. 3: Метрика accuracy для разных ядер при изменении числа признаков.

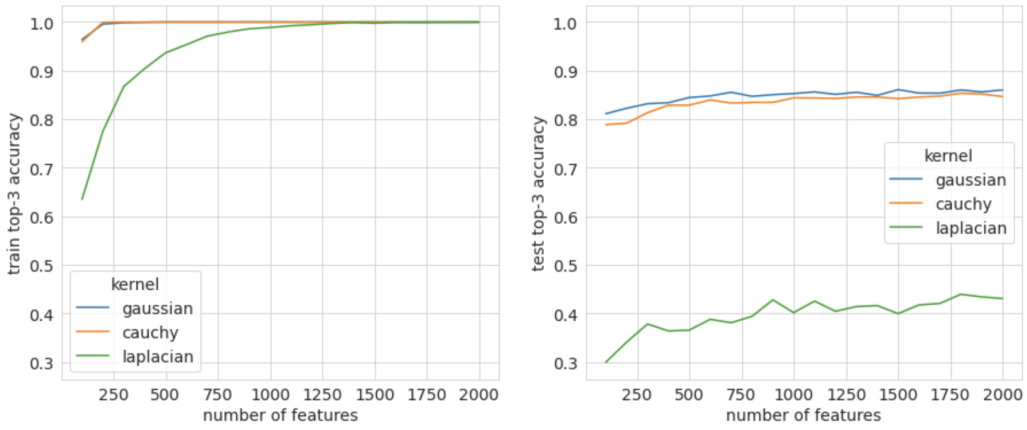

Рис. 4: Метрика top-3 accuracy для разных ядер при изменении числа признаков.

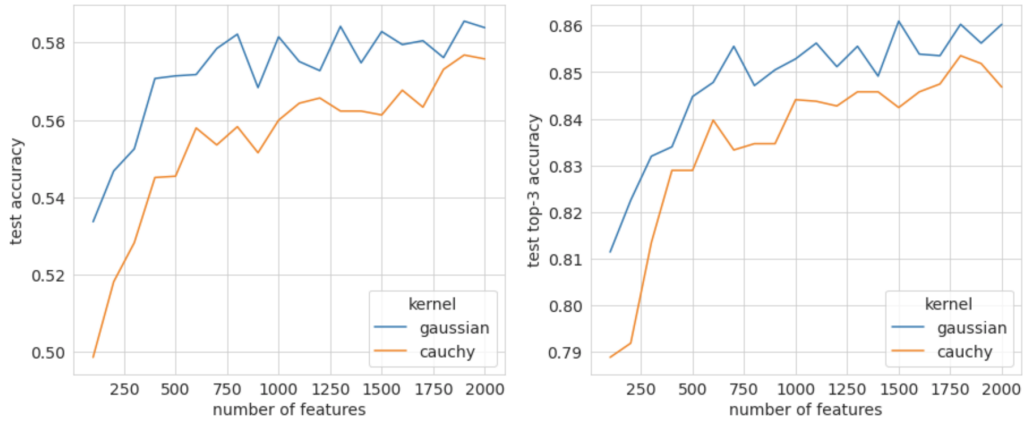

Рис. 5: Сравнение результатов для ядер Гаусса и Коши.

4.2 Ортогональные признаки

Мы провели эксперимент, который демонстрирует прирост качества при переходе от обычных случайных признаков к ортогональным. Сильнее всего этот эффект проявляется для небольшого числа признаков. Также любопытно, что при использовании ортогональных признаков понижается качество на обучающей выборки, так что ортогонализацию можно рассматривать как регуляризацию для случайных признаков.

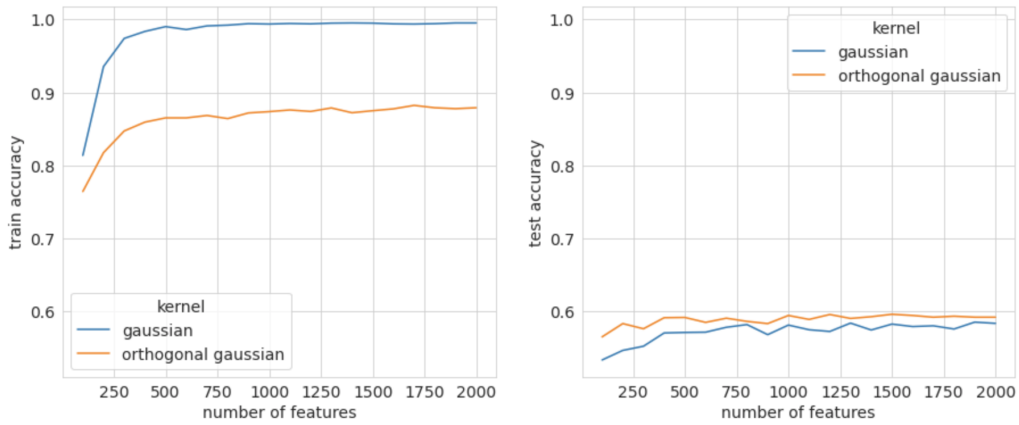

Рис. 6: Метрика accuracy для ядра Гаусса: обычные признаки против ортогональных

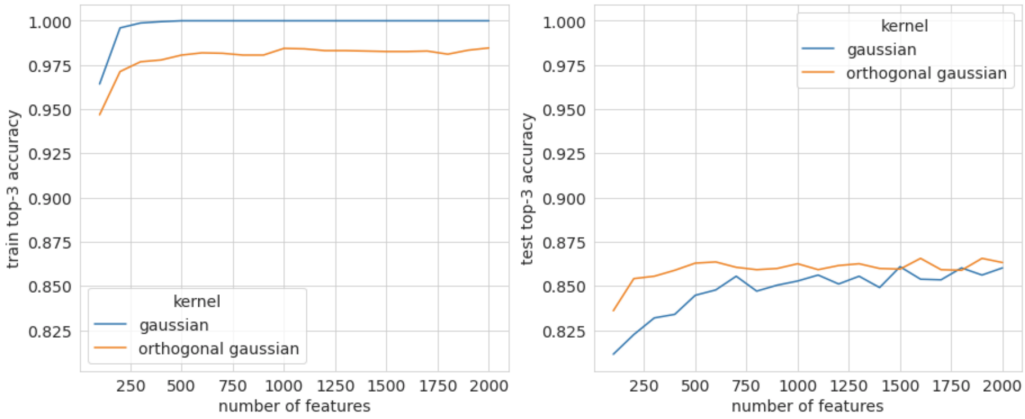

Рис. 7: Метрика top-3 accuracy для ядра Гаусса: обычные признаки против ортогональных.

4.3 Регуляризация

Еще один эксперимент показывает влияние коэффициента λ на качество модели. На графиках видно, что качество модели на обучающей выборки растет при увеличении λ, при этом максимум качества на тестовой выборке приходится не на тривиальное значение $\lambda = 1$, поэтому перебор λ как гиперпараметра имеет смысл с точки зрения увеличения качества. Эксперимент подтверждает роль λ как коэффициента регуляризации.

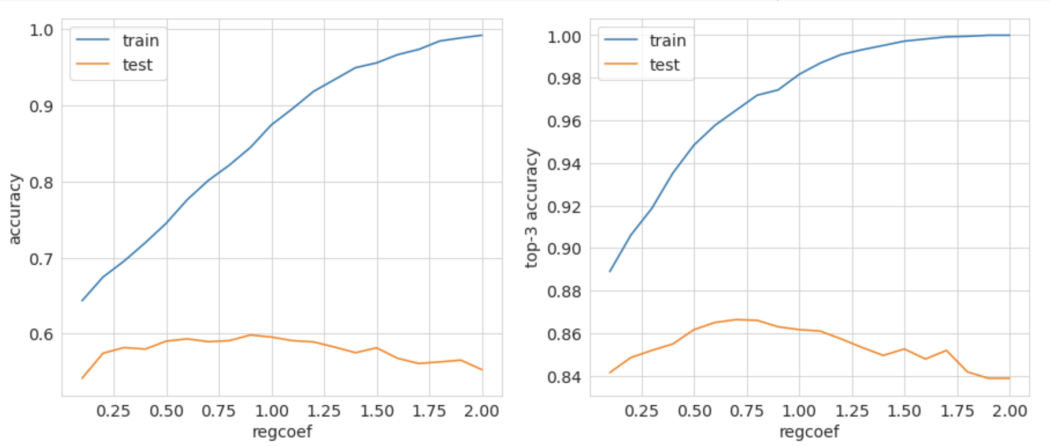

Рис. 8: Влияние коэффициента λ на качество на обучающей и тестовой выборке.

4.4 Результаты

В качестве бейзлайна мы взяли логистическую регрессию и градиентный бустинг в реализации LightGBM, которые применили к тем же самым эмбеддингам. Результаты на тестовой выборке представлены в таблице:

	LogReg	LightGBM	ORF + LogReg
accuracy	0.649	0.633	**0.674**
top-3 accuracy	0.892	0.864	**0.900**

Такого результата удалось добиться при использовании ортогональных гауссовских признаков при значениях параметров $D = 4000, \lambda = 0.85$. Стоит заметить, что результат сильно варьировался от запуска к запуску, но всегда был выше, чем у логистической регрессии. Таким образом, использование случайных признаков привело к приросту качества на 2%.

А теперь посмотрим, как не стоит использовать Random Fourier Features

5 Какие еще эмбеддинги

Зачем нам эти сложности с обучением каких-то нейронных сетей? Просто возьмем картинки и представим их в виде вектора размерности 224×224. (Обратим внимание, что канал использовался только один, 3 канала уменьшали качество и требовали больше времени и памяти). После применения к ним PCA разложения в теории должны остаться только самые важные пиксели, так что такого размера не стоит пугаться.

5.1 Опять эксперименты

Обычная логистическая регрессия к таким данным смогла добиться результата в 24%, что значительно уступает обучению на эмбеддингах.

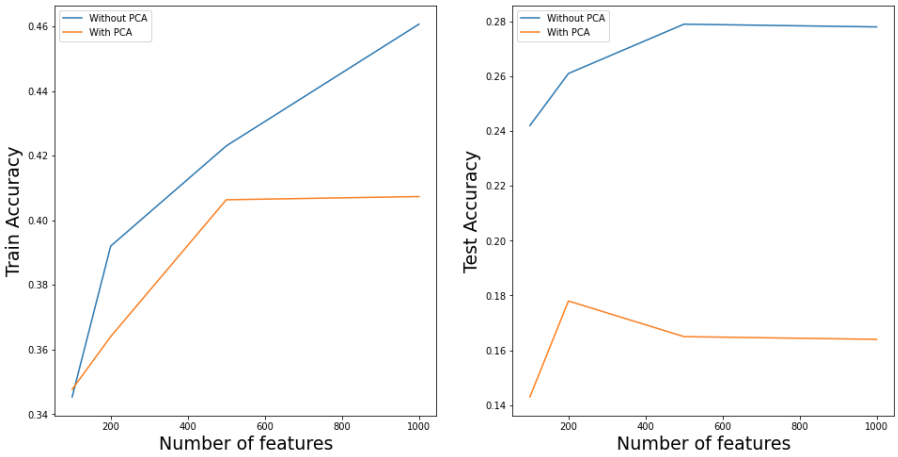

Рис. 9: Влияние числа признаков на качество на обучающей и тестовой выборке при использовании PCA с размерностью 100.

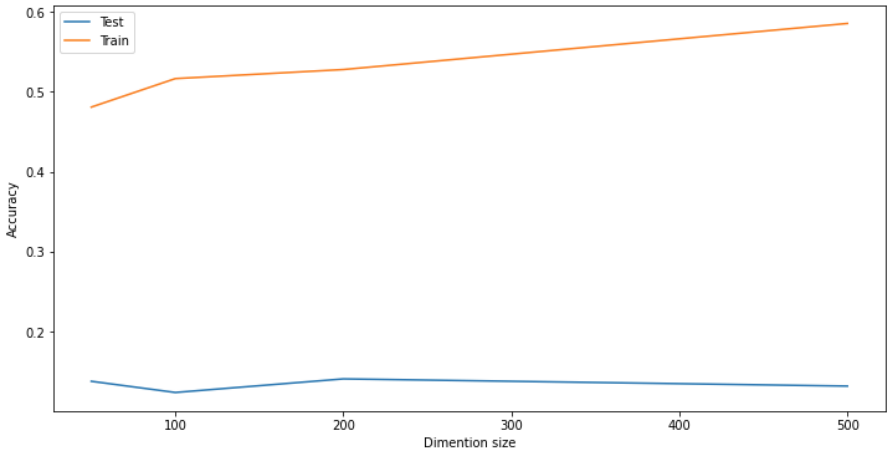

Рис. 10: Влияние размерности PCA на качество на обучающей и тестовой выборке.

Как мы видим, PCA не может дать хорошую базу признаков для RFF, поэтому качество получается посредственное даже для тренировочной выборке. На наш взгляд, дело в специфике задачи. Стиль художника практически невозможно выразить небольшим количеством пикселей, как это можно сделать в задачах классификации объектов на картинке.

Для него необходимо смотреть на всю картину, а также извлекать более сложные признаки. В качества плюса можно отметить только ускорение обучения, ведь признаков стало не 50 тысяч, а примерно 1000, но при качестве, близком к качеству случайных предсказаний, это сомнительный плюс.

Возможно, ортогональные признаки как-то смогут исправить ситуацию?

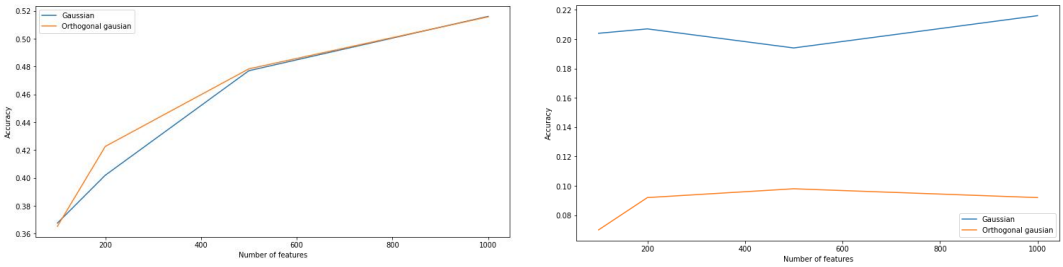

Рис. 11: Качество при использовании ортогональных признаков на тренировочной и тестовой выборках

Нескореллированные признаки это, конечно, хорошо, но когда они не очень хорошие, это не может помочь. Из забавного: ортогонализация действительно работает как регуляризация и замедляет обучение. Увы, признаки настолько плохие, что ничего не выучилось.

6 Классификация текстов

А что же у нас с классификацией текстов? Для анализа был взят датасет с заголовками статей с Medium.

6.1 Данные

Данные состояли из заголовков и подзаголовков постов, по которым необходимо было определить тематику статьи: одну из 92. Текст преобразовался с помощью tf-idf.

6.2 Эксперименты

Простой логрег выбивает на таком 43% точности. Попробуем улучшить это с помощью силы рандома.

	Gaussian	Laplacian	Cauchy
train	0.67	0.94	0.85
test	0.28	0.17	0.29

Давайте, пожалуйста, забудем про эти попытки применить RFF к текстам в таком виде. Казалось, что tf-idf может дать нормальную базу и мы получим какие-то ключевые слова с помощью PCA. Но нет, не получим, так как судя по всему у нас будет просто много слов общего смысла, которые часто употребляются. Из-за этого в большинстве случаев логрег не может сойтись, но что-то да выучивает (подозреваю, что 1-2 самых популярных класса).

Также была попытка накинуть на все это 1d свертки перед логрегом, но они все же предназначены для последовательностей, поэтому вышло еще хуже.

7 Вывод

Что можно сказать в конце. RFF — это неплохой способом добыть признаки, если есть хорошая база, а также может являться одним из видов регуляризации. Невероятных высот с помощью такого применения рандома добиться вряд ли получится, но добить последние несколько процентов точности вполне можно, если использовать этот алгоритм с умом.

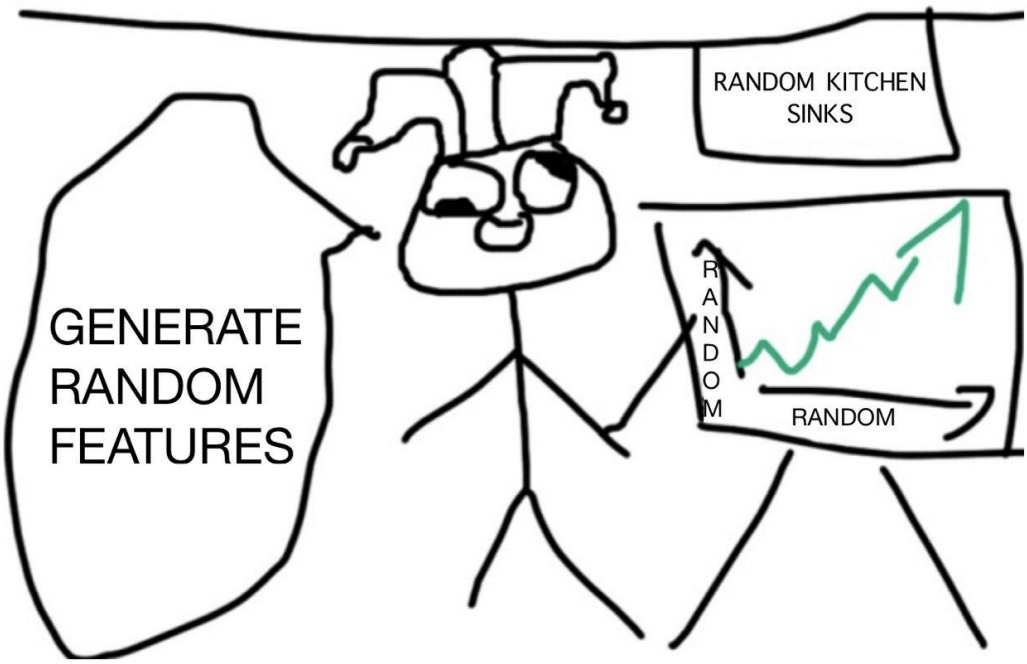

Рис. 12: Спойлер: так ничего не заработает

8 Список источников

1. *Panchajanya Banerjee (Pancham).* Impressionist classifier data. URL: https://www.kaggle.com/delayedkarma/impressionist-classifier-data, свободный (дата обращения: 03.06.2020).

2. *Nitin Viswanathan.* Artist Identification with Convolutional Neural Networks

3. *Ali Rahimi, Benjamin Recht.* Random Features for Large-Scale Kernel Machines.

4. *Felix Xinnan Yu et al.* Orthogonal Random Features.

5. *Microsoft Corporation.* LightGBM documentation. URL: https://lightgbm.readthedocs.io/en/latest, свободный (дата обращения: 04.06.2020).

6. *amrrs.* Medium post titles. URL: https://www.kaggle.com/nulldata/medium-post-titles, свободный (дата обращения: 31.05.2020).