Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Фоминская Галина
кафедра ММП, группа 317
2017

Метод опорных векторов

1 Введение

В данной работе был проведен анализ метода опорных векторов для задачи бинарной классификации. Рассмотренная реализация методов была написана на языке *Python*. Эксперименты были проведены на искусственно сгенерированных датасетах. Для решения задачи с ограничениями использовался метод внутренней точки. Для решения задачи без ограничений был реализован и исследован метод PEGASOS.

2 Эксперименты

2.1 Сравнение работы алгоритмов

В первом эксперименте сравнивается работа алгоритмов PEGASOS, полного и стохастического субградиентного спуска, а также метода внутренней точки. В таблицах 1, 2, 3, 4, 5 приведены зависимости функции потерь и времени работы от размера выборки и размерности пространства признаков для этих алгоритмов. Рассматривались следующие выборки:

1. Линейно разделимые
2. Разделимые, но нелинейно
3. Неразделимые
4. С несбалансированными классами
5. С выбросами

Из результатов видно, что все рассмотренные методы плохо минимизируют целевую функцию для нелинейно разделимых и неразделимых данных. Это можно объяснить тем, что метод пытается разделить гиперплоскостью неразделимые объекты, и остается много объектов из обучающей выборки, на которых алгоритм будет ошибаться. Они будут вносить существенный вклад в функцию потерь. Аналогичные рассуждения объясняют довольно большие значения функции потерь для выборок маленьких размерностей с выбросами. При этом качество для выборок с несбалансированными классами достаточно хорошее и примерно одинаковое для всех методов. Это объясняется тем, что большая часть объектов обучающей выборки (все, кроме опорных векторов), не влияют на решение. Также можно заметить существенный рост времени работы субградиентного спуска с ростом размеров обучающей выборки. Это и ожидалось, так как на каждой итерации этого метода происходит вычисление субградиента по всей обучающей выборке.

Таблица 1: Objective, $l = 10, d = 10$

данные	**PEGASOS**	**SubGD**	**SubSGD**	**cvxopt.qp**
линейно разделимые	0.097	0.0	0.006	0.103
нелинейно разделимые	0.779	3.505	0.824	0.824
неразделимые	0.992	0.973	1.049	0.994
несбалансированные	0.058	0.0	0.0	0.038
с выбросами	0.508	0.633	0.507	0.521

Таблица 2: Objective, $l = 100, d = 100$

данные	**PEGASOS**	**SubGD**	**SubSGD**	**cvxopt.qp**
линейно разделимые	0.01	0.0	0.0	0.01
нелинейно разделимые	0.794	39.5	0.994	0.758
неразделимые	0.988	0.985	1.159	0.991
несбалансированные	0.004	0.0	0.253	0.004
с выбросами	0.211	0.643	0.246	0.212

Таблица 3: Objective, $l = 1000, d = 1000$

данные	**PEGASOS**	**SubGD**	**SubSGD**	**cvxopt.qp**
линейно разделимые	0.001	0.0	0.001	0.001
нелинейно разделимые	0.837	312.923	0.739	0.758
неразделимые	0.994	0.989	1.03	0.989
несбалансированные	0.0	0.0	0.085	0.0
с выбросами	0.203	5.641	0.242	0.201

Таблица 4: Время (с), $l = 10, d = 10$

данные	**PEGASOS**	**SubGD**	**SubSGD**	**cvxopt.qp**
линейно разделимые	0.018	0.003	0.001	0.008
нелинейно разделимые	0.012	0.746	0.001	0.005
неразделимые	0.013	0.007	0.003	0.004
несбалансированные	0.017	0.003	0.001	0.006
с выбросами	0.016	0.694	0.002	0.005

Таблица 5: Время (с), $l = 100, d = 100$

данные	**PEGASOS**	**SubGD**	**SubSGD**	**cvxopt.qp**
линейно разделимые	0.012	0.005	0.001	0.029
нелинейно разделимые	0.013	1.818	0.002	0.022
неразделимые	0.013	0.042	0.011	0.02
несбалансированные	0.012	0.007	0.001	0.034
с выбросами	0.013	1.861	0.001	0.04

Таблица 6: Время (с), $l = 1000, d = 1000$

данные	**PEGASOS**	**SubGD**	**SubSGD**	**cvxopt.qp**
линейно разделимые	0.174	0.401	0.004	7.552
нелинейно разделимые	0.152	98.691	0.01	4.754
неразделимые	0.159	0.032	0.062	3.433
несбалансированные	0.126	0.964	0.003	6.35
с выбросами	0.125	77.506	0.004	6.118

2.2 Сравнение работы алгоритма SVM с полиномиальным и RBF ядрами

Аналогично исследуем целевую функцию и время работы SVM с ядрами в зависимости от особенностей данных. Результаты приведены в таблицах 8, 9, 10, 11, 12, 13. Видно, что оба метода одинаково хорошо минимизируют функционал и для линейно разделимых выборок, и для нелинейно разделимых. При этом время обучения модели с RBF ядром сильно возрастает с увеличением размеров выборки и размерности пространства признаков. Это объясняется необходимостью вычислять и хранить какое-то время матрицу попарных расстояний между объектами обучающей выборки.

Таблица 7: $l = 1000, d = 2$

	Linear		**RBF**	
данные	**accuracy**	**objective**	**accuracy**	**objective**
линейно разделимые	1.0	-0.47	1.0	-2.063
нелинейно разделимые	0.496	-0.802	1.0	-5.126
неразделимые	0.456	-0.971	0.456	-9.7
несбалансированные	1.0	-0.416	1.0	-2.029
с выбросами	0.896	-0.644	0.896	-3.991

2.3 Точность классификации

Сравним теперь точность работы рассмотренных методов. В таблице 7 приведены значения точности для SVM с линейным ядром и с RBF ядром, и соответствующие значения целевой функции. В данном случае решалась двойственная задача, поэтому целевая функция максимизировалась. Как и ожидалось, оба метода с точностью 1.0 классифицируют линейно разделимые объекты, а метод RBF с той же точностью классифицирует и нелинейно разделимые. При этом видно, что для линейной модели большим значениям целевой функции соответствуют большие значения accuracy. Это же наблюдается и для модели с RBF ядром.

2.4 Настройка параметров

У метода SVM есть параметр C. Также у моделей с полиномиальным и RBF ядрами есть параметры gamma и degree. С помощью кросс-валидации исследуем, как эти параметры влияют на качество работы модели. Лучшие результаты метод с RBF ядром показал при $C = 10$ и $C = 1000$, метод с полиномиальным при $C = 1000$. При всех значениях C плохо разделимые данные одинаково плохо классифицируются($average\_accuracy = 0.5$). При этом линейно разделимые и нелинейно разделимые данные одинаково хорошо классифицируются при указанных C ($average\_accuracy = 1.0$). В дальнейшем использовались $C = 10$ для RBF и $C = 1000$ для полиномиального. Для RBF ядра лучшими значениями gamma оказались 1, 5, 10, они дают одинаково хорошее качество на разделимых данных. Для полиномиального ядра лучшими значениями degree оказались 3 и 5.

Таблица 8: Objective, $l = 10, d = 10$

данные	**Polynomial**	**RBF**
линейно разделимые	-0.049	-3.144
нелинейно разделимые	-3.345	-5.139
неразделимые	-2.98	-6.475
несбалансированные	-0.05	-2.797
с выбросами	-1.619	-6.06

Таблица 9: Objective, $l = 100, d = 100$

данные	**Polynomial**	**RBF**
линейно разделимые	-0.005	-50.0
нелинейно разделимые	-0.132	-50.0
неразделимые	-0.894	-50.0
несбалансированные	-0.005	-42.0
с выбросами	-0.161	-50.0

Таблица 10: Objective, $l = 1000, d = 1000$

данные	**Polynomial**	**RBF**
линейно разделимые	-0.0	-500.0
нелинейно разделимые	-0.013	-500.0
неразделимые	-0.08	-500.0
несбалансированные	-0.0	-420.0
с выбросами	-0.017	-500.0

Таблица 11: Время (с), $l = 10, d = 10$

данные	**Polynomial**	**RBF**
линейно разделимые	0.01	0.01
нелинейно разделимые	0.008	0.009
неразделимые	0.007	0.008
несбалансированные	0.009	0.011
с выбросами	0.006	0.015

Таблица 12: Время (с), $l = 100, d = 100$

данные	**Polynomial**	**RBF**
линейно разделимые	0.022	0.026
нелинейно разделимые	0.026	0.021
неразделимые	0.025	0.021
несбалансированные	0.025	0.025
с выбросами	0.023	0.025

Таблица 13: Время (с), $l = 1000, d = 1000$

данные	**Polynomial**	**RBF**
линейно разделимые	1.351	111.045
нелинейно разделимые	1.739	96.117
неразделимые	1.797	102.44
несбалансированные	1.487	92.865
с выбросами	1.598	94.943

2.5 Сравнение стратегий выбора шага

Исследуем зависимость времени работы и качества предсказаний на отложенной выборке (accuracy) от величины шага для субградиентных полного и стохастического методов. Максимальное значение accuracy для полного метода получилось равным 0.928 при значениях $step\_alpha = 1, step\_beta = 2$. Время работы составило 24.6 мс. Для стохастического метода лучшее accuracy равно 0.908, достигается при тех же значениях параметров шага. Время работы стохастического метода в этом случае равно 8.38 мс. Сравним результаты с точностью и временем алгоритма PEGASOS. Время его работы на этих же данных составило 16.9 мс при $accuracy = 0.916$.

2.6 Зависимость от размера подвыборки

В таблице 14 приведены время и качество на отложенной выборке для стохастического субградиентного спуска для различных значений $batch\_size$. Видно, что в целом с увеличением размера подвыборки точность возрастает, а время работы не сильно меняется.

Таблица 14: $l = 1000, d = 2$

$batch\_size$	**Время (мс)**	**accuracy**
1	9.18	0.908
10	20.5	0.912
50	18.9	0.928
100	17.6	0.928

2.7 Сравнение SVM и логистической регрессии

Рассмотрим альтернативный метод решения задачи классификации — логистическую регрессию. В отличие от нее, у метода SVM есть ряд достоинств, в частности:

1. Возможность перехода в спрямляющее пространство и, как следствие, возможность решать задачи для нелинейно разделимых данных.

2. Предсказания метода SVM зависят только от опорных векторов, что позволяет решать ряд задач даже с небольшими выборками и сохранять при этом хорошую обобщающую способность.

2.8 Визуализация для двумерного случая

На рисунке 1 изображены рассматриваемые выборки с двумя признаками. На рисунках 2, 3 изображены разделяющие поверхности для линейного ядра и для RBF ядра. На рисунках 4, 5 изображены крупными точками опорные векторы. Видно, что прямая хорошо разделяет линейно разделимые данные, линейно разделимые данные с выбросами и с несбалансированными классами. Так как на решение влияют только опорные векторы, несбалансированность классов не влияет на качество работы алгоритма. На рисунке 6 изображены разделяющие поверхности для линейно разделимых данных для различных значений параметра gamma RBF ядра. Видно, что при увеличении значения gamma алгоритм настраивается на один из классов и теряет свою обобщающую способность.

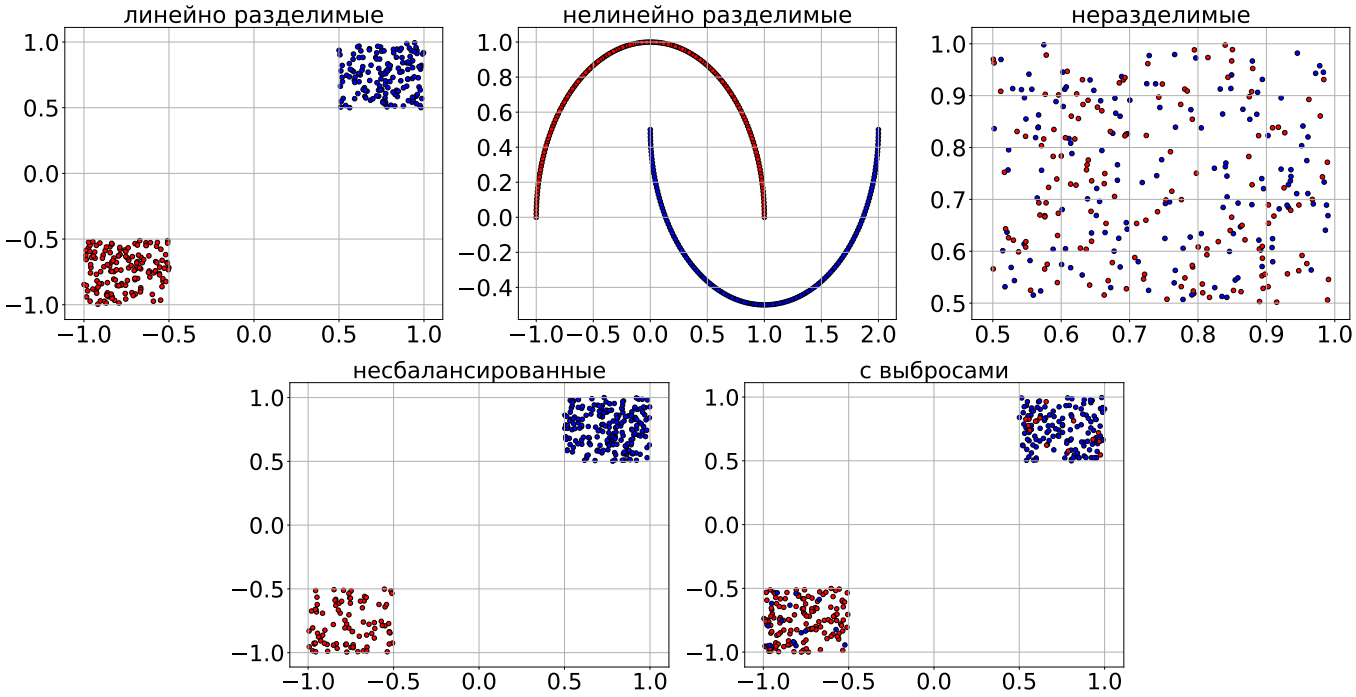

Рис. 1: Выборки

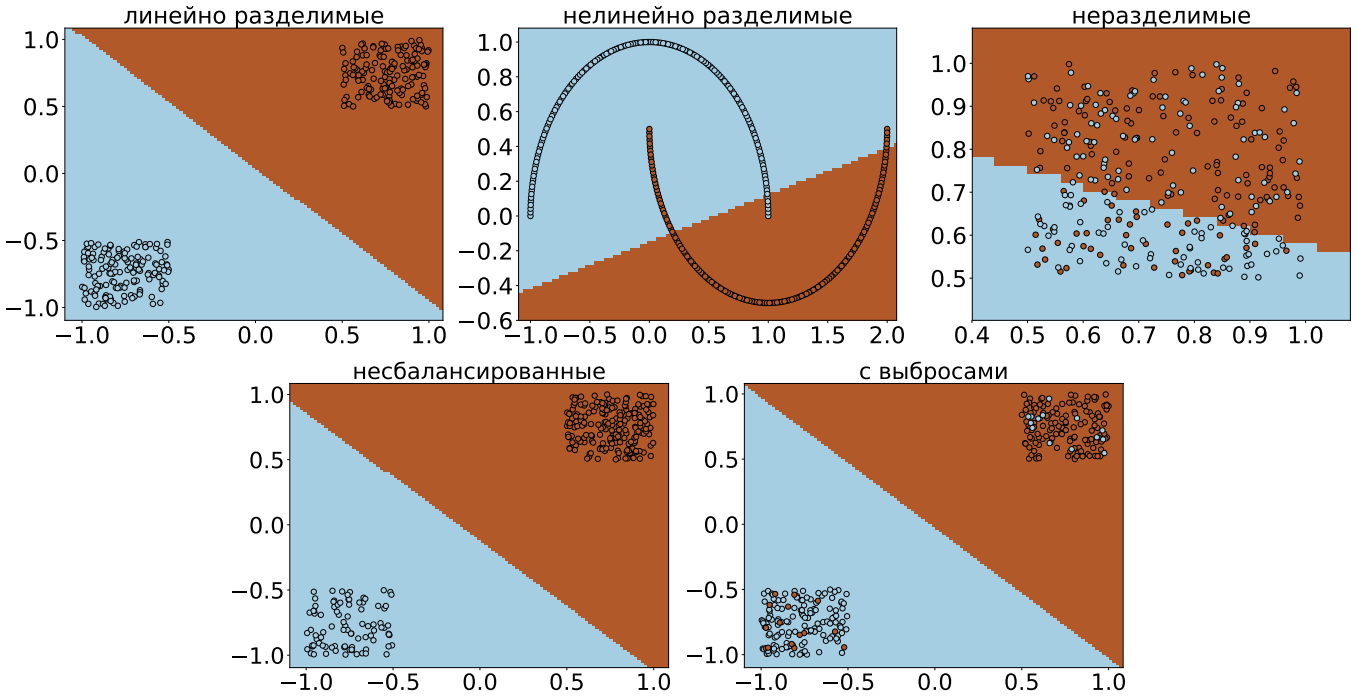

Рис. 2: Разделяющие поверхности для линейного ядра

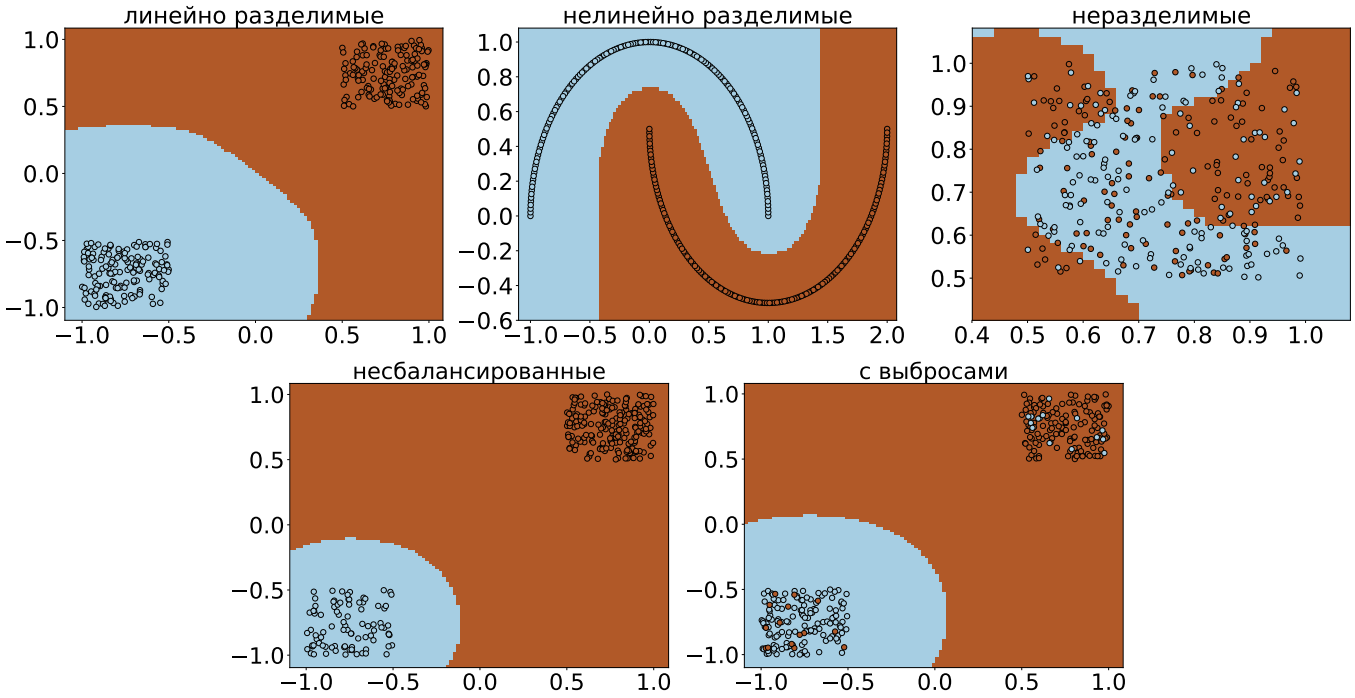

Рис. 3: Разделяющие поверхности для rbf ядра

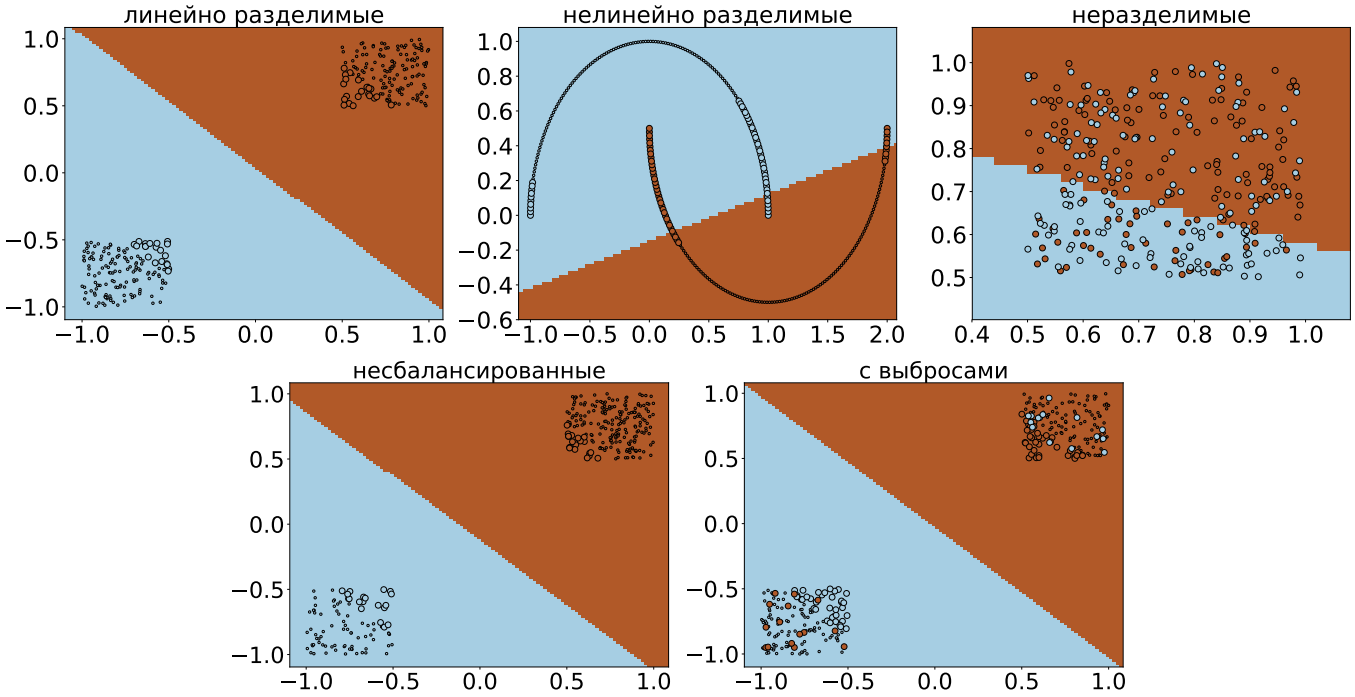

Рис. 4: Опорные векторы для линейного ядра

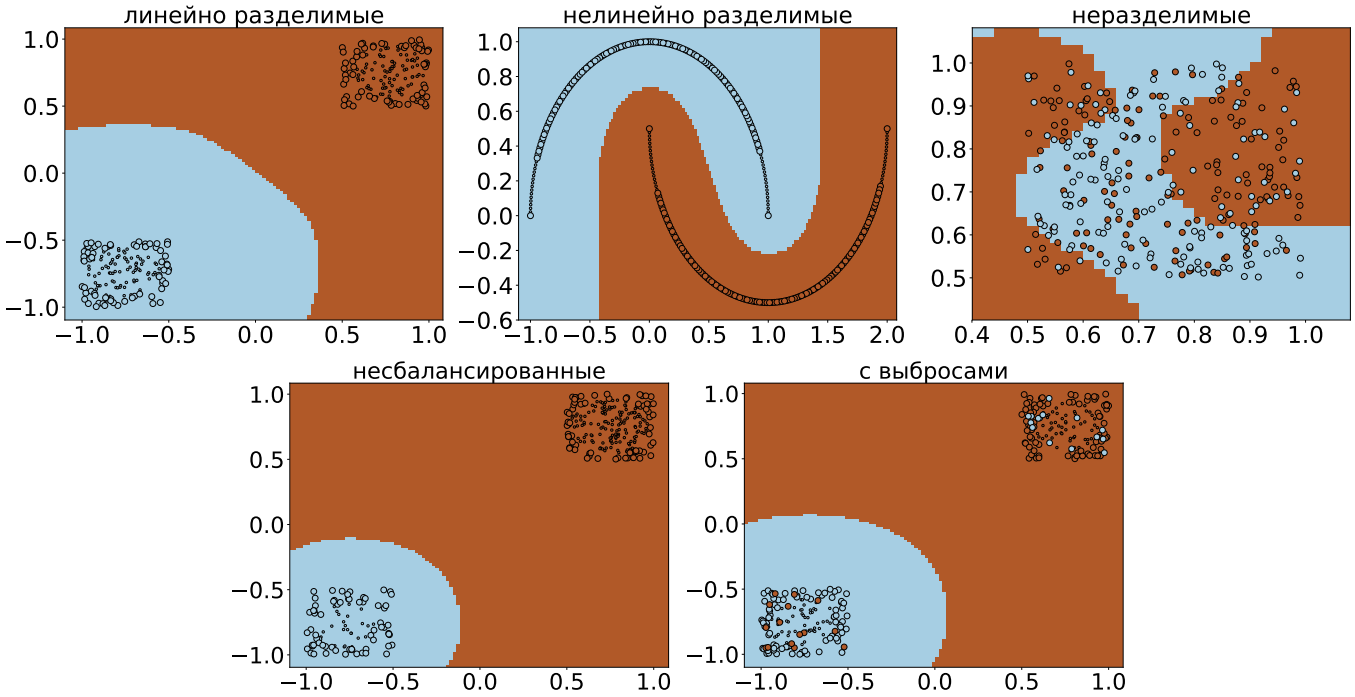

Рис. 5: Опорные векторы для rbf ядра

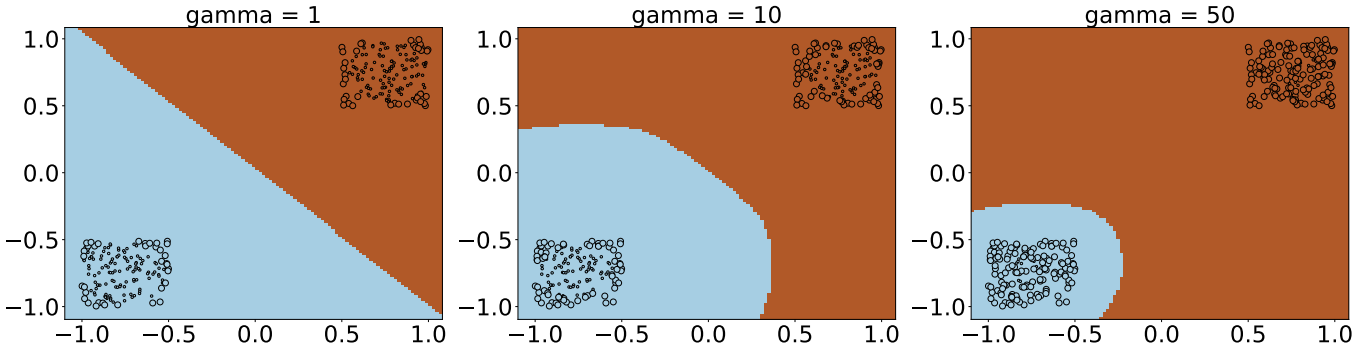

Рис. 6: Зависимость от gamma