The Kernel Trick и его приложения в машинном обучении

Н. В. Павличенко, Н. С. Торопцев, Д. А. Павлов

Московский физико-технический институт
pavlichenko.nv@phystech.edu

20 декабря 2019 г.

Аннотация

В данной статье расматривается применение ядерных метов в машинном обучении. Дается краткое описание метода опорных векторов, спрямляющих подпространств и ядер. В работе были рассмотрены три основных ядра: полиномиальное, ядро всех подмножеств и гауссовское ядро. Для всех методов приводится теоретическое обоснование, сами методы сравниваются как на сгенерированных данных, так и на реальных данных, таких как Boston Housing Prices. Помимо этого рассматривается эффективность применения спрямляющих подпространств в методах, основанных на решающих деревьях.

I. Введение

Линейные методы использовались с самого зарождения таких наук как статистика и машинное обучение. Они действительно хороши с теоретической точки зрения: для них доказано много теорем, найдены доверительные интервалы для отклика в различных моделях и, более того, существует аналитическое решение для нахождения оптимальных параметров алгоритма. Однако они имеют существенные проблемы: они очень плохо работают при нелинейных зависимостях в данных. Собственно, хотелось бы иметь метод, который позволял бы использовать все достоинства линейных моделей и, при этом, который позволял бы приближать и нелинейные зависимости определенных видов. Идея ядерных методов заключается в том, что пространство признаков, в котором зависимость нелинейная, можно отобразить в другое пространство, в котором она уже будет линейной. Это пространство называется спрямляющим. При этом на самом деле, достаточно знать только как выражается скалярное произведение в новом пространстве, что мы и будем называть функцией ядра. Далее рассмотрим теоретический аспект подробнее.

II. Задача линейной классификации

Мы будем рассматривать задачу классификации объектов исходных данных. Каждый объект будет отнесён в класс 1 или 2 в зависимости от своих свойств, что очевидно мотивировано в задачах машинного обучения. Более конкретно, для заданного набора $\{x\}_{i=1}^{n} \subset X$ мы хотим построить классификатор

$$a : X \to \{0, 1\}.$$

Понятно, что для реальных задач оправдано только $X = \mathbb{R}^n$, то есть классифицировать мы будем численные объекты с некоторым множеством признаков.

III. Метод опорных векторов

Учитывая огромное количество алгоритмов и свойств, полученных для линейно соотносящихся объектов, наиболее выгодно было бы получить искомый классификатор в виде гиперплоскости в $\mathbb{R}^n$, явно разделяющей

объекты на два нужных класса. Тогда в случае существования такой плоскости, называемом случаем линейной разделимости, искомый классификатор будет представим в виде

$$a(x) = \text{sign}(\langle \mathbf{w} \cdot x \rangle - b), \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}.$$

Геометрический смысл состоит в том, что гиперплоскость можно однозначно задать перпендикуляром из нуля и единичным перпендикуляром к плоскости, после чего произведения этого перпендикуляра на объекты будет давать положительное или отрицательное число в зависимости от того полупространства, в котором находится объект (с точностью до переноса всех объектов на $-\mathbf{b}$).

В случае линейной разделимости нам интересно найти такую плоскость, которая будет наиболее отстоять от всех объектов для наибольшей эффективности при дальнейшей классификации. Переформулируем задачу: найдём такую плоскость, для которой существуют две плоскости a_1, a_2 параллельные ей с классификаторами $a_1(x), a_2(x)$ такими, что

$$\rho(a_1, a_2) \to \max,$$

$$\nexists a_1(x) = -a_2(x)$$

(второе выражение означает отсутствие объектов между плоскостями). Так как расстояние между параллельными плоскостями выражается как $\frac{2}{\|\mathbf{w}\|}$, а второе выражение можно переписать в явном виде классификаторов (нормируя расстояние между плоскостями к 2), получим оптимизационную задачу поиска такой плоскости:

$$\begin{cases} \|\mathbf{w}\| \to \min, \\ c_i(\langle \mathbf{w} \cdot x_i \rangle - b) \geq 1, \\ c_i \in \{-1, 1\} \end{cases}$$

Для данной задачи выполнена сильная двойственность, и в силу условий Каруша-Куна-Таккера ей эквивалентна задача

$$\begin{cases} \frac{1}{2}\|\mathbf{w}\|^2 - \sum \lambda_i (c_i(\langle \mathbf{w} \cdot x_i \rangle - b) - 1) \\ \qquad \to \min_{w,b}, \max_{\lambda}, \\ \lambda_i \geq 0, \\ c_i \in \{-1, 1\}. \end{cases}$$

Последнюю задачу сведём к двойственной задаче программирования:

$$\begin{cases} \sum \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j c_i c_j \langle x_i, x_j \rangle \to \max_{\lambda}, \\ \lambda_i \geq 0, \\ c_i \in \{-1, 1\} \\ \sum \lambda_i c_i = 0. \end{cases}$$

Для которой существует решение в виде

$$\begin{cases} \mathbf{w} = \sum \lambda_i c_i x_i, \\ b = w \cdot x_i - c_i (\forall \lambda_i \geq 0). \end{cases}$$

Тогда после решения задачи искомый классификатор будет записан в виде

$$a(x) = \text{sign}(\sum \lambda_i c_i \langle x_i, x \rangle - b)$$

.

IV. Kernel trick и представление через спрямляющее пространство

Понятно, что не всегда на реальных данных выполнено свойство линейной разделимости. Однако иногда можно ввести новый признак, благодаря которому в новом пространстве большей размерности может быть выполнено свойство линейной разделимости (скажем, классы на плоскости разделены некоторой окружностью, тогда введя новый признак удалённости от центра этой окружности, объекты будут линейно разделимы по проекции на этот признак и любой из двух первоначальных). Такое новое пространство признаков назовём спрямляющим.

Рассмотрим более подробно такой перевод из первоначального пространства признаков, который осуществляется функцией ϕ. Заметимтот важный факт, что в новом пространстве классификатор, найденный изложенным ранее методом, будет зависеть только от скалярных произведений объектов в новом пространстве:

$$a(y) = \text{sign}(\sum \lambda_i c_i \langle y_i, y \rangle - b).$$

Введём функцию ядра:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle.$$

Составленную из них матрицу назовём ядром:

$$K = (\langle \phi(x_i), \phi(x_i) \rangle)_{i,j}.$$

Таким образом, можно увидеть два способа задания ядра: явный (через конкретный вид ϕ) и неявный (через задание $k(x_i, x_j)$). Заметим, что искомый классификатор в спрямляющем пространстве однозначно выражается через неё. Таким образом, новые классификаторы можно строить задавая ядра различными способами, а так же изучая их свойства.

Алгоритм работы нового метода следующий: получаем скалярные произведения в новом пространстве используя явный или неявный вид ядра, решаем оптимизационную задачу квадратичного программирования (максимизируем функционал, используя, например, градиентный спуск), и по решению строим классификатор в явном виде указанном выше.

V. Свойства ядер

Рассмотрим несколько заключений о свойствах ядер. Значимость этих свойств состоит в возможности неявного построения ядер без указания явного вида отображения, а задавая саму матрицу скалярных произведений с выполнением некоторых свойств.

Теорема 1. K - ядро тогда и только тогда, когда она положительно полуопределена и симметрична.

Доказательство аналогично такому же, приведённому для свойств матрицы Грамма. Следствие состоит в том, что приведя удовлетворяющую на заданных данных функцию $k(x_i, x_j)$, можно утверждать, что неявное отображение будет являться ядром.

Теорема 2. Линейные комбинации с неотрицательными коэффциентами и произведения ядер так же будут являться ядрами.

Доказательство тривиально следует из предыдущей теоремы.

Теорема 3. Пусть мы имеем последовательность ядер, сходящуюся в каждой точке x, z. Тогда

$$K(x, z) = \lim_{n \to \infty} K_n(x, z)$$

также будет являться ядром.

Данное свойство доказывается через свойство предела отграниченности от нуля и теорему 1.

Теорема 4. Нормированное ядро, то есть

$$\hat{K}(x, z) = \frac{K(x, z)}{\sqrt{K(x, x)K(z, z)}}$$

является ядром.

Доказательство. Для ядра K существует отображение ϕ. Рассмотрим отображение $\hat{\phi}(x) = \frac{\phi(x)}{\|\phi(x)\|}$. Тогда

$$\hat{K}(x, z) = (\langle \hat{\phi}(x_i), \hat{\phi}(x_j) \rangle)_{i,j},$$

так как

$$\frac{K(x, z)}{\sqrt{K(x, x)K(z, z)}} = \frac{\langle \phi(x), \phi(z) \rangle}{\|\phi(x)\| \|\phi(z)\|} = \langle \hat{\phi}(x), \hat{\phi}(z) \rangle.$$

□

VI. Некоторые важные примеры

Примеры построения неявных ядер, основанных на фактах предыдущего пункта.

i. Полиномиальное ядро

Теорема 5. Для многочлена $p(x)$ с неотрицательными коэффициентами

$$K(x, z) = p(\langle x, z \rangle)$$

является ядром.

Доказательство. По определению, $\langle x, z \rangle$ является ядром (достаточно взять отображение $\phi(x) = x$). Далее, $(\langle x, z \rangle)^n$ является ядром как произведение ядер. Линейная комбинация ядер с неотрицательными коэффициентами является ядром, ч. т. д. □

На практике важен следующий вид полиномиального ядра с весовым коэффициентом:

$$K(x,z) = (\langle x,z\rangle - R)^m, R \in \mathbb{R}.$$

Можно показать, что данный коэффициент контролирует отношения весов малых и больших степеней произведений объектов в неявном отображении. В самом деле, в явном виде

$$K(x,z) = \sum \binom{m}{k} R^{m-k} (\langle x,z\rangle)^k,$$

откуда можно показать, что отношение веса монома степени k к моному первой степени равно $\frac{1}{R^{k-2}}$.

ii. Гауссовское ядро

Теорема 6.

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

является ядром.

Доказательство. Заметим, что

$$K(x,z) = \exp\left(\frac{\langle x,z\rangle}{\sigma^2}\right)$$

является ядром как предел сходящейся всюду последовательности ядер (из разложения в ряд Тейлора следует, что экспонента представима как предел многочленов с неортицательными коэффициентами, по предыдущему пункту мы знаем, что последние являются ядрами). Нормированное ядро также является ядром по теореме выше. Тогда ядром является

$$\frac{\exp\left(\frac{\langle x,z\rangle}{\sigma^2}\right)}{\sqrt{\exp\left(\frac{\langle x,x\rangle}{\sigma^2}\right)\exp\left(\frac{\langle z,z\rangle}{\sigma^2}\right)}}$$

$$= \exp\left(\frac{\langle x,z\rangle}{\sigma^2} - \frac{\langle x,x\rangle}{2\sigma^2} - \frac{\langle z,z\rangle}{2\sigma^2}\right) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) = K(x,z). \quad \square$$

Заметим, что σ аналогично является весовым коэффициентом для сравнения вкладов малых и больших степеней в значение функции ядра. С помощью изменения значения параметра можно контролировать вклад малых степеней и недопускать переобучения модели.

Гауссовские ядра являются самыми распространёнными в использующихся моделях обучения в связи с хорошей изученностью в смежных сферах, а так же их близостью к линейным моделям. Действительно, если исходные данные являются гауссовскими векторами и рассматривается некоторая линейная модель, то разность векторов выражает близкую к гауссовскому вектору с нулевым матожиданием. Таким образом, в случае независимости построенный классификатор будет отражать функцию риска для отклонения, а для зависимой выборки может служить мерой корреляции двух классов.

VII. Решение оптимизационной задачи для нахождения весов

Как было показано ранее, задача обучения сводится к решению следующей оптимизационной задачи:

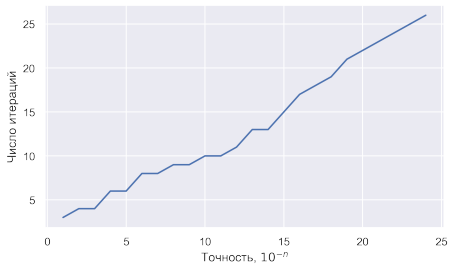
$$\begin{cases} \sum \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j c_i c_j \langle x_i, x_j\rangle \to \max_\lambda, \\ \lambda_i \geq 0, \\ c_i \in \{-1, 1\} \\ \sum \lambda_i c_i = 0. \end{cases}$$

Сегодня одним из наиболее применяемых для этих целей является метод проекции градиента. Этот метод представляет из себя по сути обобщение метода градиентного спуска для задачи оптимизации с ограничениями. Алгоритм заключается в поиске минимума функционала вдоль кусочно линейного пути. Каждая следующая итерация алгортима записывается в виде

$$x_{k+1} = P[x_k - \alpha \Delta f(x_k)].$$

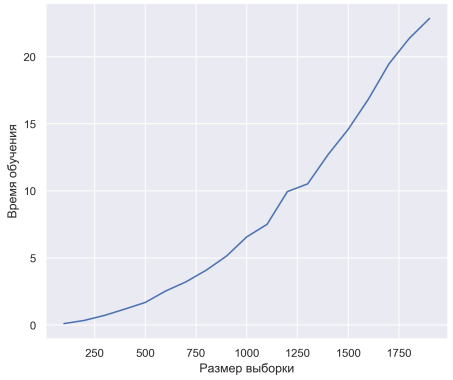
В данной работе использовалось уменьшение шага спуска в два раза на каждой итерации. Алгоритм продолжал работу пока изменение отрезок между старой и новой точкой имел норму, превосходящую 10^{-15}. Для того, чтобы оценить эффективность данного алгоритма, рассмотрим зависимость числа итераций от требуемой точности. Как мы видим, для достижения даже очень большой точности,

Рис. 1: Число итераций метода проекции градиента для достижения требуемой точности



такой как 10^{-25}, алгоритму требуется не более 30 шагов. Осталось рассмотреть зависимость времени работы алгоритма от размера выборики. Как мы видим, зависимость линей-

Рис. 2: Время работы метода проекции градиента в секундах в зависимости от размера обучающей выборки



ная. Теперь рассмотрим применение метода проекции градиента для задачи оптимизации метода опорных векторов. Сгенерируем линейно разделимую выборку и решим задачу квадратичного программирования SVM. Для нахождения оптимальных весов. Для обучения потребовалось 15 итераций алгоритма. Как мы видим, полученные веса задают оптимальную разделяющую прямую для данного набора точек. Теперь рассмотрим линейно неразделимую выборку следующего вида. Теперь решим задачу оптимизации, где вместо скалярного произведения будем исользовать функцию полиномиального ядра. Для проверки результатов воспользуемся полиномиальным отображением и убедимся, что алгоритм нашел веса, задающие оптимальную гиперплоскость в новом пространстве. Как мы видим, метод сошелся и для линейно неразделимой выборки, причем в новом пространстве алгоритм нашел наиболее оптимальную разделяющую гиперплоскость. Таким образом, метод проекции градиента применим для задачи обучения машины опорных векторов. Далее рассмотрим качество получаемых алгоритмах в различных задачах классификации и регрессии.

Рис. 3: Результат обучения SVM методом проекции градинента

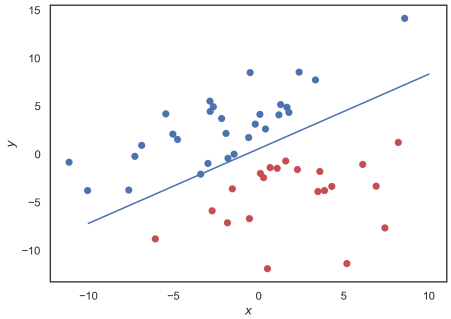


Рис. 4: Линейно неразделимая выборка

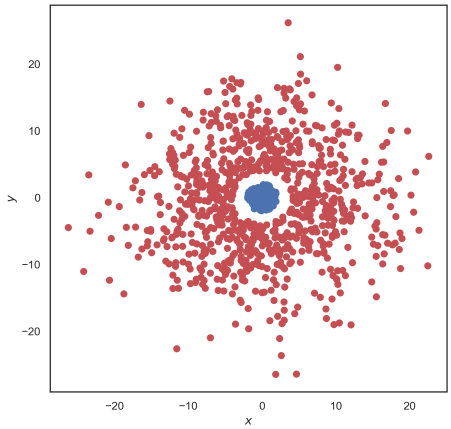
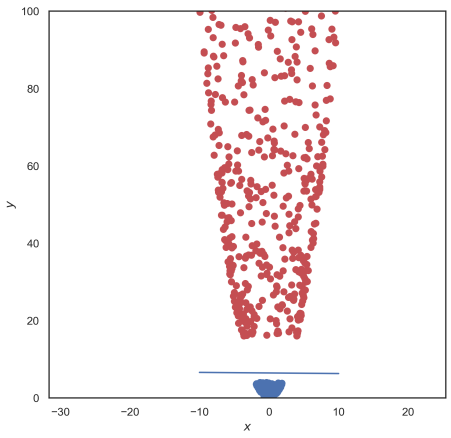


Рис. 5: Линейно неразделимая выборка в новом пространстве

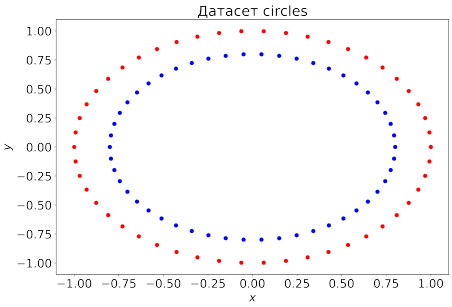


VIII. Тестирование на синтетических данных

i. Классический SVM

Рассмотрим задачу классификации на сгенерированном датасете circles из библиотеки sklearn. Этот датасет предстаялет из себя два круга — их точки представляют, собственно, два класса. Круги вложены друг в друга, то есть, в обычном евклидовом пространстве выборка не является линейно разделимой.

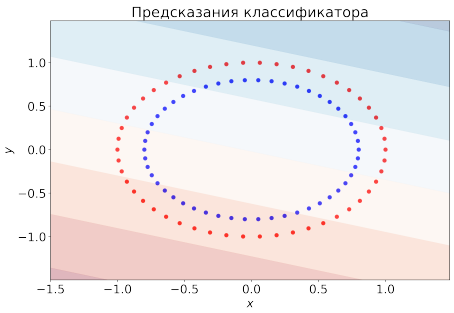
Рис. 6: Классы объектов датасета circles



Посмотрим как ведет себя логистическая регрессия при на этих данных.

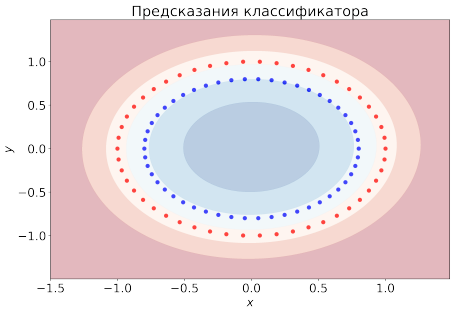
На рис. 7 мы можем наблюдать, что классификатор не нашел никакой зависимости в данных, и разделил выборку некоторой, достаточно случайной гиперплоскостью, так как

Рис. 7: Предсказания линейного классификатора



при нескольких запусках одного и того же алгоритма градиентного спуска, при разных начальных весах будут получаться разные разделяющие гиперплоскости. Accuracy score такого алгоритма составляет 0.44, что означает, что он имеет качество не лучше константного алгоритма. Теперь рассмотрим предсказания алгоритма, обученного на данных, преобразованных квадратичным отображением.

Рис. 8: Предсказания линейного классификатора с квадратичным ядром



Как и ожидалось, качество такого алгоритма намного выше, accuracy score составляет 1.0. Это объясняется тем, что после преобразования признаков, алгоритм может использовать разделяющие поверхности вида $x < r$, что задает круг, как это и предполагается в данных.

ii. Нелинейные алгоритмы

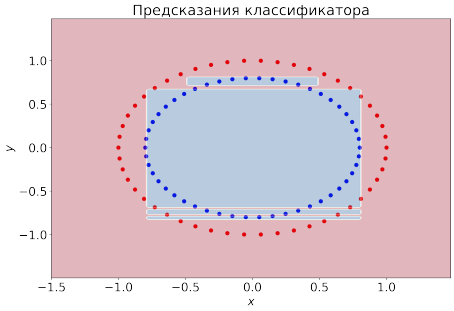
По сути, использование квадратичного преобразования признаков представляет из себя

отображение некоторой функцией

$$\varphi : \mathbb{R}^d \to \mathbb{R}^k.$$

Так как пространство, в которое происходит отображение признаков, тоже является вещественным, можно переформулировать задачу обучения в нем и использовать нелинейные алгоритмы, например, основанные на решающих деревьях. Для начала обучим решающее дерево для задачи классификации на данных из датасета circles. На рис. 8 мы можем отме-

Рис. 9: Предсказания решающего дерева

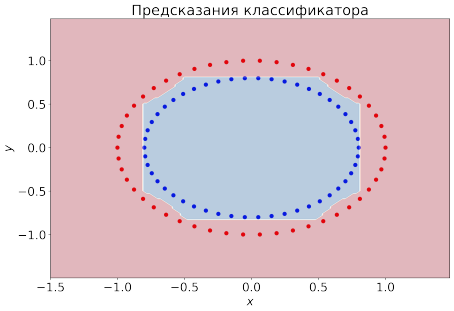


тить несколько моментов:

- Решающее дерево может строить только ортогональные разбиения, то есть результат все равно неизбежно не будет идеальным.
- Мы не получили качество, как у линейной классификации с квадратичным ядром, несмотря на усложнение модели.
- Дерево явно переобучилось, что можно понять по полосам снизу, хотя это и следовало ожидать от такого неустойчивого алгоритма.

Accuracy score при этом 0.88, что довольно хорошо, но все же не идеально. Попробуем теперь обучить решающее дерево на преобразованных признаках. Здесь мы наблюдаем другую картину, разбиения теперь не только ортогональные, так как у нас есть признаки вида $x_1 \cdot x_2$. При этом разделяющая поверхность стала сильно ближе к реальной, то есть к кругу. Accuracy score тоже стал равен 1.0. То есть, переход в спрямляющее подпространство дал улучшение качества и на нелинейных моделях.

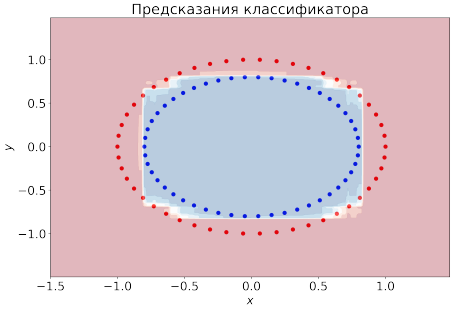
Рис. 10: Предсказания решающего дерева, обученного на квадратичных признаках



iii. Применение в композициях алгоритмов

Рассмотрим применение этого же отображения для сильных алгоритмов, позволяющих искать сложные нелинейные зависимости, например, градиентный бустинг. Будем использовать библиотеку с открытым исходным кодом Yandex CatBoost. Посмотрим на результат обучения градиентного бустинга на все том же датасете circles. Для обучения использовались параметры по умолчанию, то есть была обучена 1000 деревьев. Мы можем пронаблюдать, что результат луч-

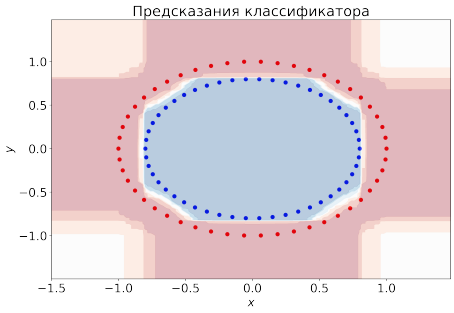
Рис. 11: Предсказания алгоритма градиентного бустинга



ше, чем у решающего дерева, хотя все же $accuracy = 0.92 < 1$, несмотря на то, что сама зависимость достаточно простая. Стоит отметить, что градиентный бустинг, являясь достаточно мощным алгоитмом, все равно строит только ортогональные разбиения, так как использует решающие деревья. Теперь

отобразим признаки функцией φ и обучим алгоритм в новом пространстве. Здесь мы на-

Рис. 12: Предсказания алгоритма градиентного бустинга

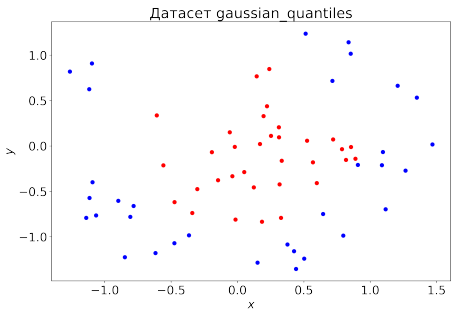


блюдаем существенное улучшение качества, *accuracy* = 1.0. Таким образом, можно сделать вывод, что использование спрямляющих подпространст может быть полезно при использовании даже в нелинейных алгоритмах.

IX. Применение гауссовского ядра в задаче классификации

Аналогично предыдущему разделу рассмотрим применение уже гауссовского ядра. Будем использовать синтетический датасет *gaussian_quantiles* из библиотеки Sci-Kit Learn. Заметим, что, как и в случае с кругами,

Рис. 13: Выборка, сгенерированная функцией make_gaussian_quantiles

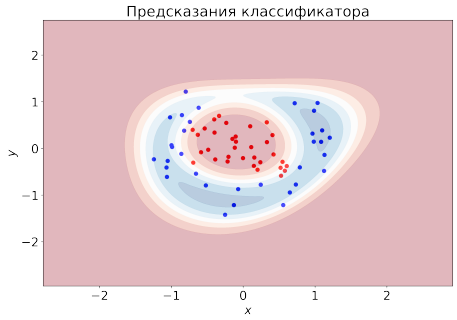


выборка не является линейно разделимой, но визуально видно, что существует явная разделяющая поверхность. Теперь, аналогично, обучим алгоритм метода опорных векторов с обычным линейным ядром и посмотрим на построенную им разделяющую гиперплоскость. Как и ожидалось, построенная разделяющая

Рис. 14: Предсказания SVM с линейным ядром

гиперплоскость не соответствует зависимости в данных. *accuracy* = 0.12, что конечно же, очень мало для такой достаточно простой задачи. Теперь обучим SVM-классификатор с гауссовским ядром. Accuracy получившегося

Рис. 15: Предсказания SVM с гауссовским ядром



алгоритма составляет 0.92, что на порядок лучше, чем у модели с линейным ядром. Как можно пронаблюдать на рис. 15, разделяющая поверхность получилось гладкой и визуально выглядит похожей на соответствующую реальной зависимости. Стоит отметить, что гауссовское ядро является одним из наиболее универсальных из возможных ядер, так как его можно использовать даже в том случае, когда мы не обладаем какими-либо априорными знаниями о характере зависимости в данных.

X. Заключение

В данной работе мы рассмотрели задачи классификации и регресси в машинном обучении. Рассмотрели метод опорных векторов как их возможное решение, поставили задачу оптимизации и предложили алгоритм ее решения для обучения метода. Результаты показали, что метод применим для получения хорошего качества как на синтетических данных, так и на реальных. Кроме того, идеи перевода признаков в спрямляющее пространство применимы не только для линейных алгоритмов, но и для более сложных моделей.

Список литературы

[Shawe-Taylor, J., Cristianini, N., 2004] Shawe-Taylor, J., Cristianini, N. (2004). Kernel Methods for Pattern Analysis. Cambridge University Press

[S. Boyd, L. Vandenberghe, 2009] S. Boyd, L. Vandenberghe (2009). Convex Optimization. Cambridge University Press