

Машинное обучение

Метод опорных векторов

(support vector machines)

Д.Ю. Хартьян

Метод опорных векторов

- Метод опорных векторов (support vector machines) - это более сложный алгоритм, чем KNN, но всё начинается с простого вопроса:
 - Можем ли мы провести гиперплоскость, которая хорошо разделит классы друг от друга?
- Чтобы ответить на этот вопрос, давайте сначала взглянем на историю развития метода опорных векторов.

Метод опорных векторов – История

- Метод опорных векторов (support vector machines) – один из относительно недавно изобретённых алгоритмов машинного обучения, который мы изучим.
- Давайте кратко взглянем на его историю...

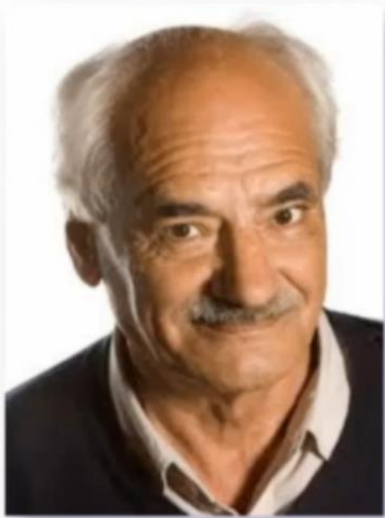
Метод опорных векторов – История

- 1960-е годы: Владимир Вапник защитил кандидатскую диссертацию в Институте проблем управления (Москва)
- Работал в этом институте 1961-1990



Метод опорных векторов – История

- 1963: Владимир Вапник и Алексей Червоненкис опубликовали “метод обобщённого портрета” для анализа изображений компьютерами

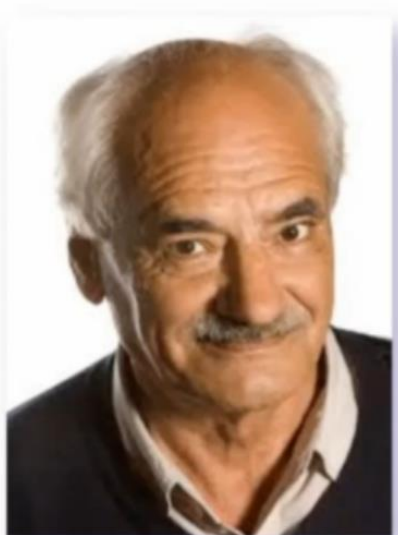


Метод опорных векторов – История

- 1964: М.А.Айзерман, М.М.Браверман, Л.И.Розоноэр публикуют “Теоретические основы метода потенциальных функций в задаче об обучении автоматов разделению входных ситуаций на классы”
- Даётся геометрическая интерпретация ядер (kernels) как произведений в пространстве признаков.

Метод опорных векторов – История

- 1974: Вапник и Червоненкис продолжают работу, публикуют книгу “Теория распознавания образов”



Метод опорных векторов – История

- 1960е-1990е года: Вапник продолжает работу над дальнейшим развитием метода опорных векторов



Метод опорных векторов – История

- 1992: Бернхард Бозер, Изабель Гюйон и Владимир Вапник предлагают нелинейный классификатор, применяя так называемый **kernel trick** для гиперплоскостей с максимальным зазором.



Метод опорных векторов – История

- 1995: Коринна Кортес и Владимир Вапник публикуют вариант метода опорных векторов с мягким зазором (soft margin)



Метод опорных векторов – История

- 1996: Владимир Вапник, Харрис Друкер, Кристофер Бургес, Линда Кауфман и Александр Смола публикуют работу “Support Vector Regression Machines”, расширяя диапазон применения SVM за пределы задач классификации.

Метод опорных векторов – История

- Чтобы получить ту версию метода опорных векторов, которую мы будем использовать, потребовалось более 30 лет!
- Сейчас продолжаются работы по увеличению производительности, а также теоретические исследования.

Метод опорных векторов – История

- Далее мы посмотрим теорию метода, постепенно продвигаясь вперед.
- Начнём с классификаторов с максимальным зазором (maximum margin classifiers), затем посмотрим классификатор опорных векторов и, наконец, сам метод опорных векторов.

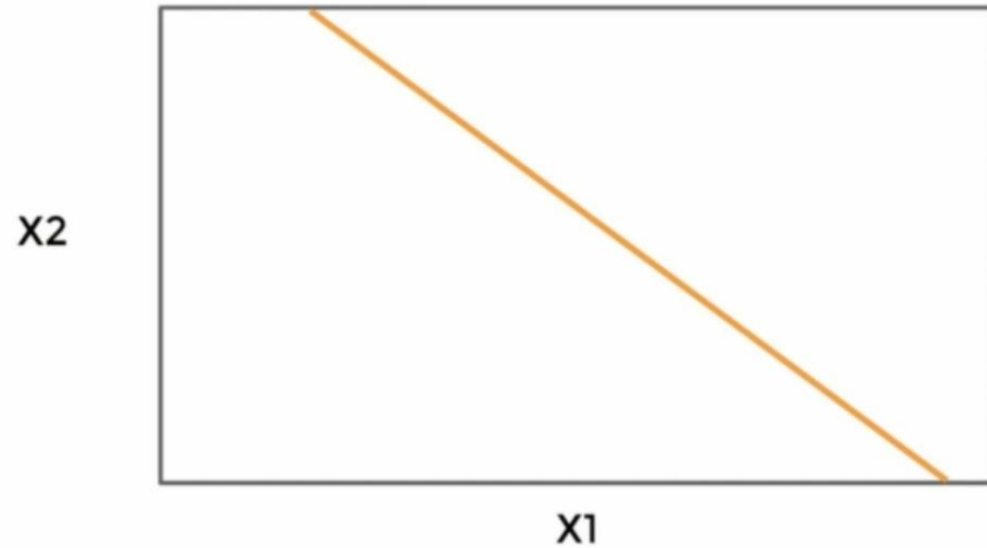
Метод опорных векторов «kernel trick» математика

- Давайте посмотрим на общие принципы математики метода SVM, и как они соотносятся с командами Scikit-Learn.
- Мы начнём с обзора классификаторов с зазорами, и как их можно описать с помощью уравнений.

Метод опорных векторов «kernel trick» математика

- Определение гиперплоскости

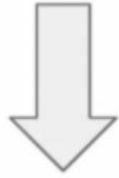
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$



Метод опорных векторов «kernel trick» математика

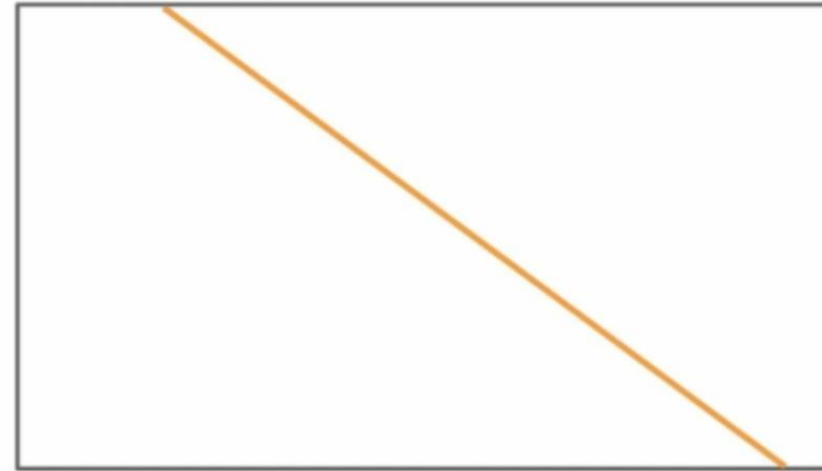
- Определение гиперплоскости

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$



$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$$

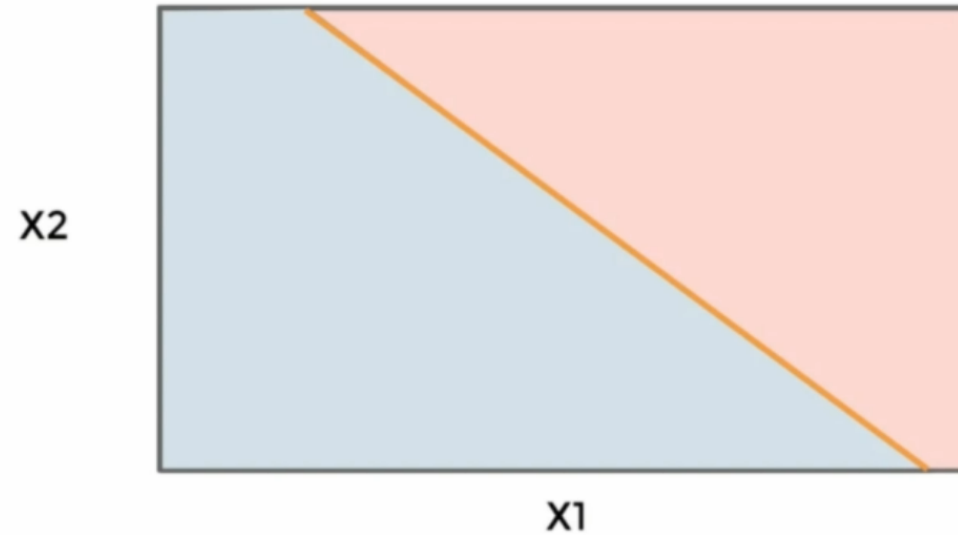
x2



x1

Метод опорных векторов «kernel trick» математика

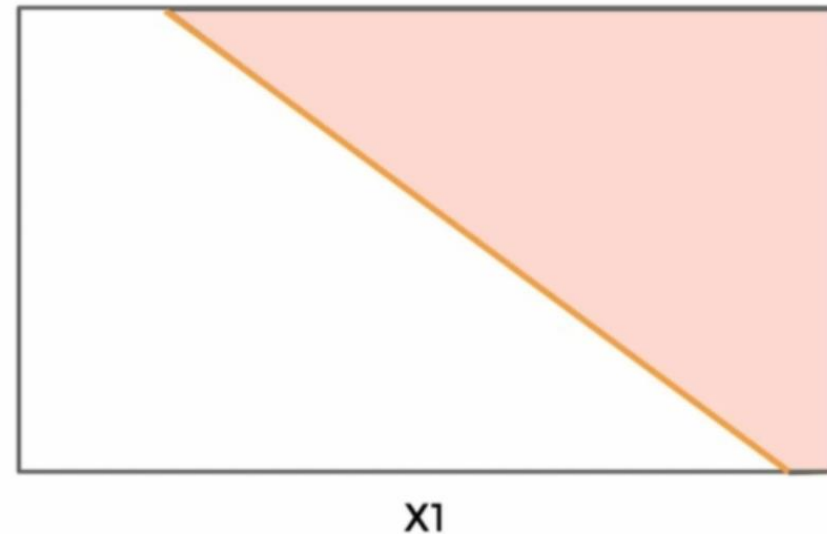
- Разделение пространства гиперплоскостью



Метод опорных векторов «kernel trick» математика

- Разделение пространства гиперплоскостью

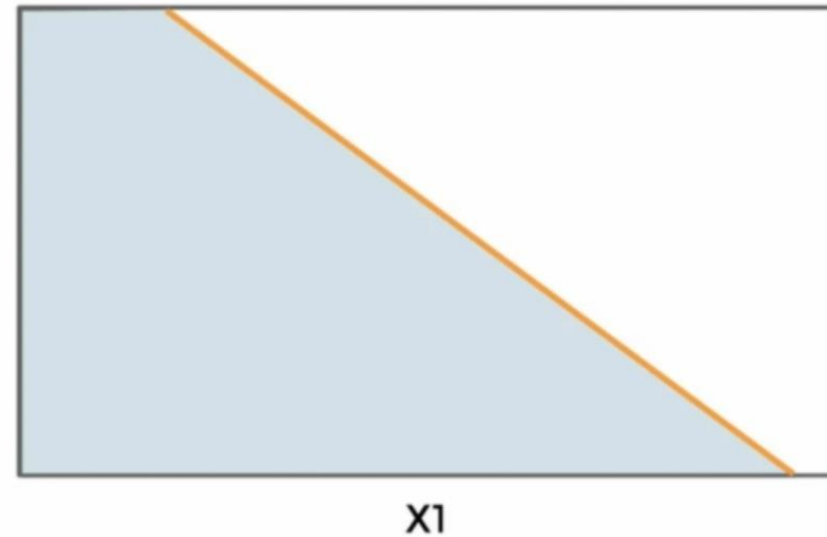
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0$$



Метод опорных векторов «kernel trick» математика

- Разделение пространства гиперплоскостью

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p < 0$$



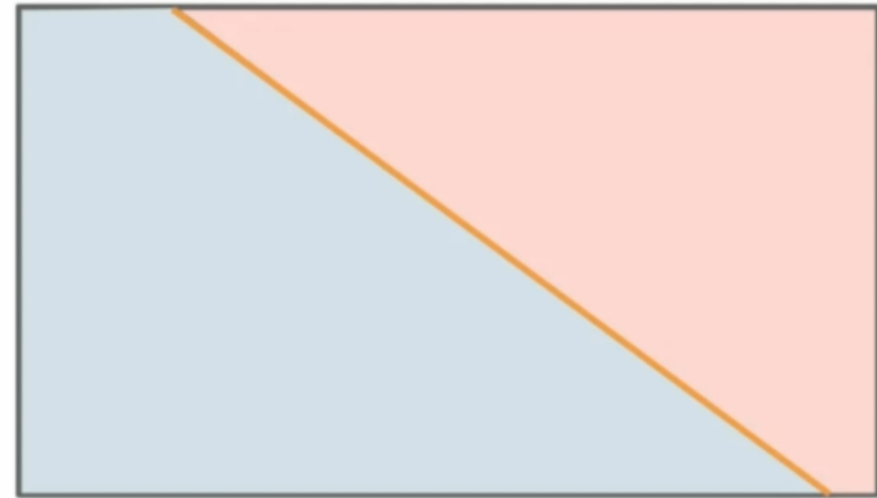
Метод опорных векторов «kernel trick» математика

- Разделение пространства гиперплоскостью

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p < 0$$

x2

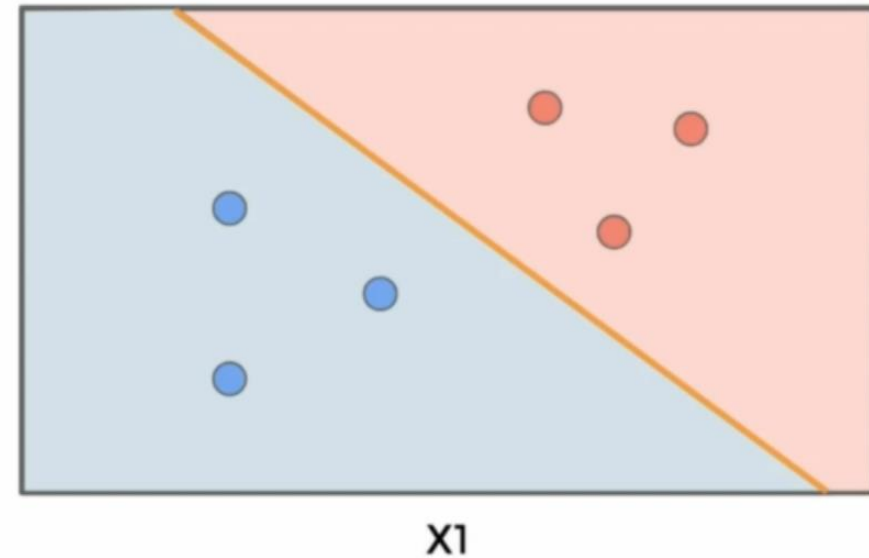


x1

Метод опорных векторов «kernel trick» математика

- Точки с данными

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix} \quad x_2$$



(1..n) – количество точек

(1..p) – количество признаков

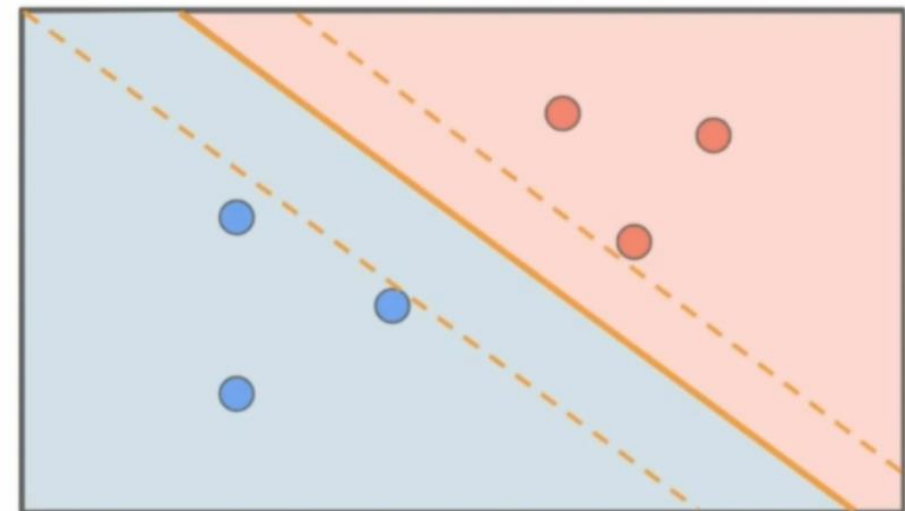
Метод опорных векторов «kernel trick» математика

- Классификатор максимального зазора

Max Margin
Classifier

максимизируем M
 $\beta_0, \beta_1, \dots, \beta_p, M$

x2



x1

Метод опорных векторов «kernel trick» математика

- Классификатор максимального зазора

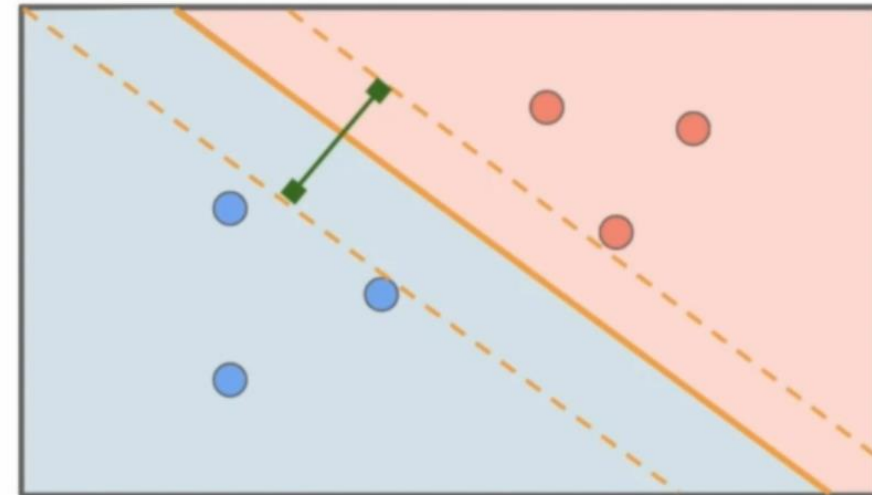
Max Margin
Classifier

максимизируем

$\beta_0, \beta_1, \dots, \beta_p, M$

M

x2



x1

Метод опорных векторов «kernel trick» математика

- Классификатор максимального зазора

Max Margin
Classifier

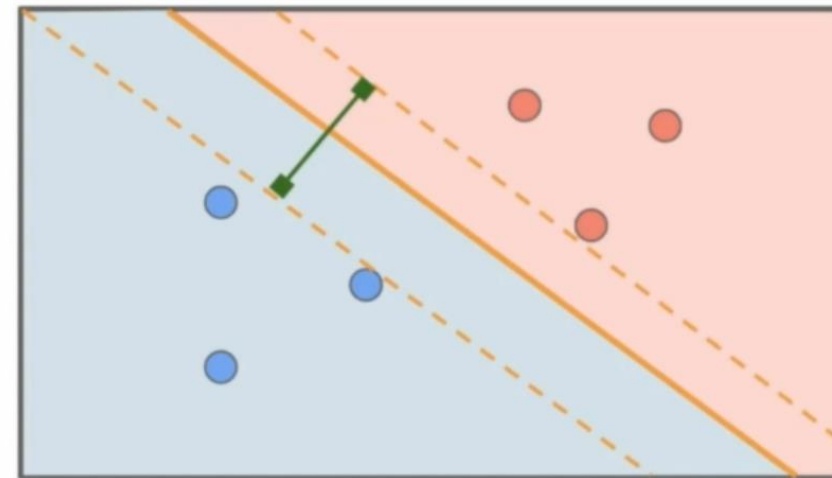
максимизируем

$\beta_0, \beta_1, \dots, \beta_p, M$

при условии $\sum_{j=1}^p \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$

x2



x1

Метод опорных векторов «kernel trick» математика

- Классификатор максимального зазора

Max Margin
Classifier

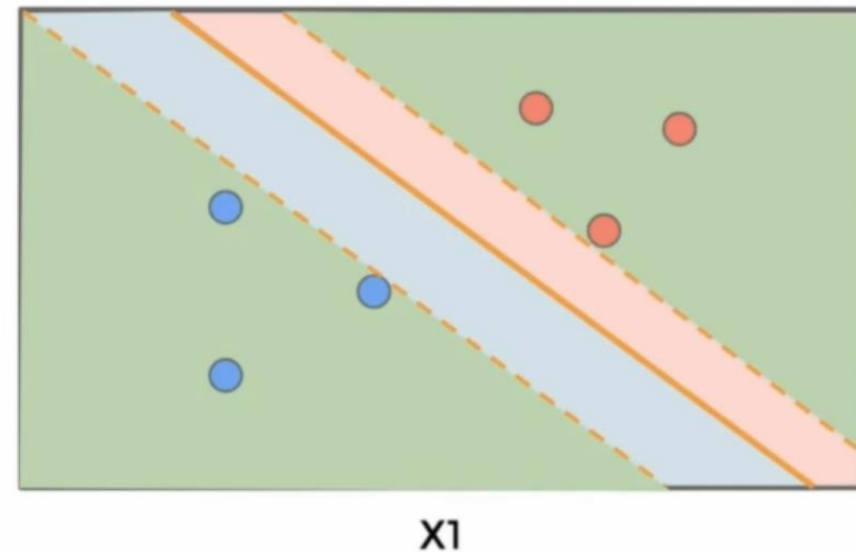
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

максимизируем M
 $\beta_0, \beta_1, \dots, \beta_p, M$

при условии $\sum_{j=1}^p \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$

x2



Метод опорных векторов «kernel trick» математика

Max Margin Classifier

- Классификатор максимального зазора

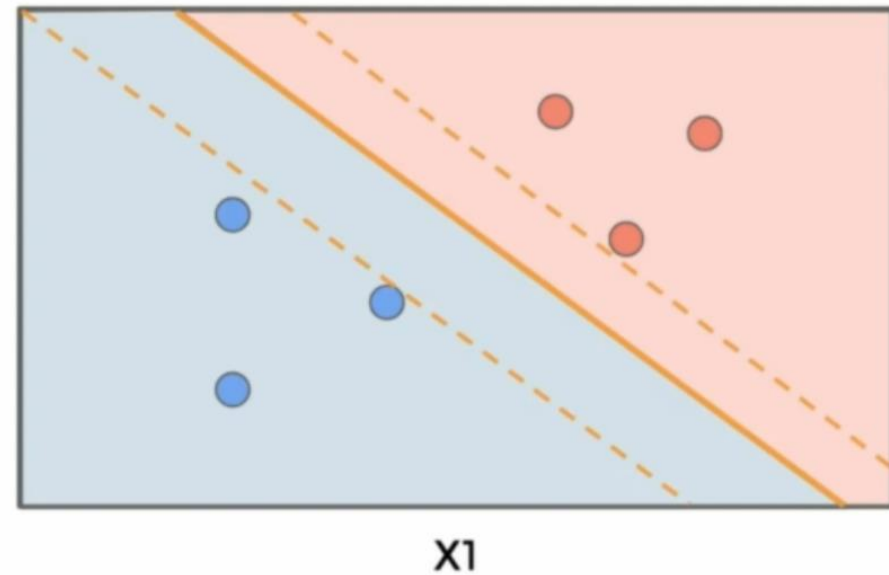
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

максимизируем M
 $\beta_0, \beta_1, \dots, \beta_p, M$

при условии $\sum_{j=1}^p \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$

x2



Метод опорных векторов «kernel trick» математика

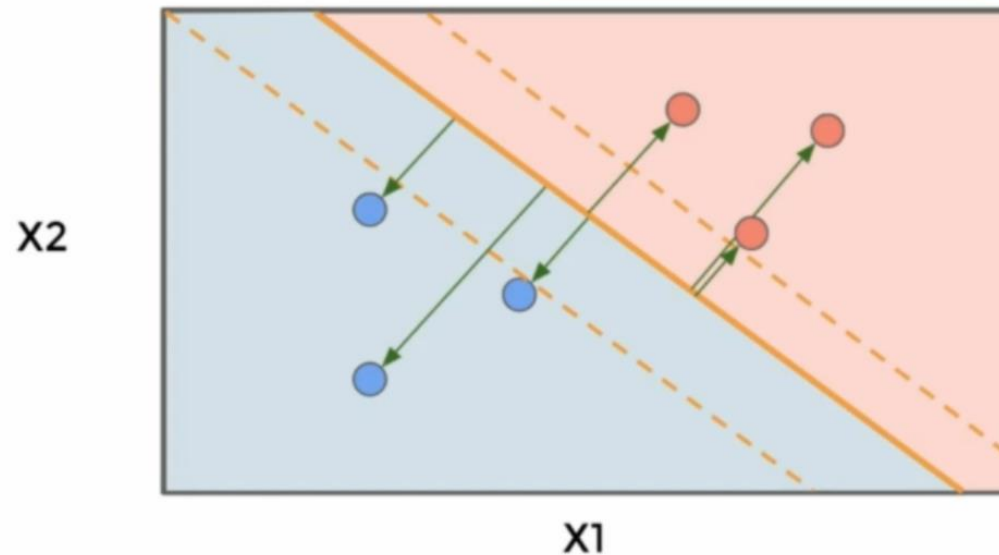
- Классификатор максимального зазора

Max Margin Classifier

при условии

$$\sum_{j=1}^p \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$



Метод опорных векторов «kernel trick» математика

- Классификатор максимального зазора

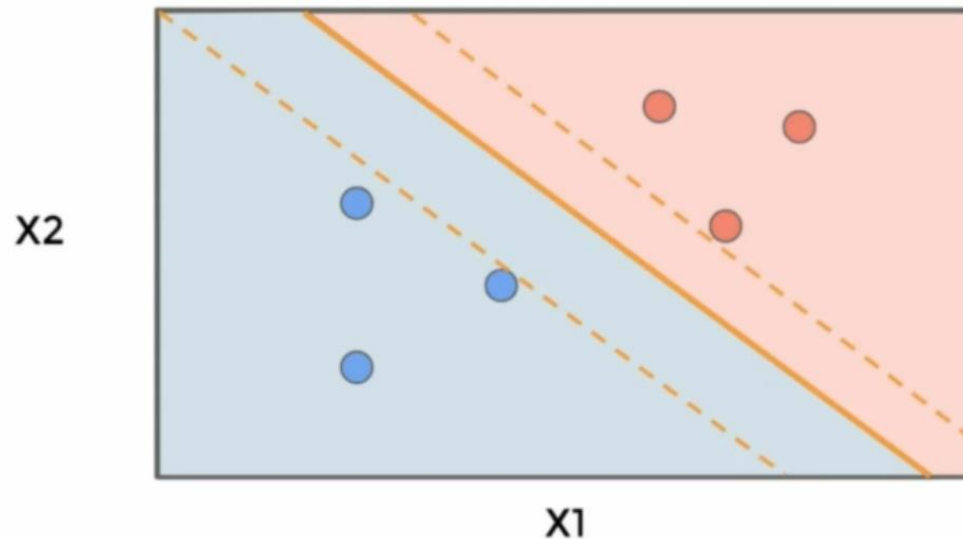
Max Margin Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

максимизируем M
 $\beta_0, \beta_1, \dots, \beta_p, M$

при условии $\sum_{j=1}^p \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$

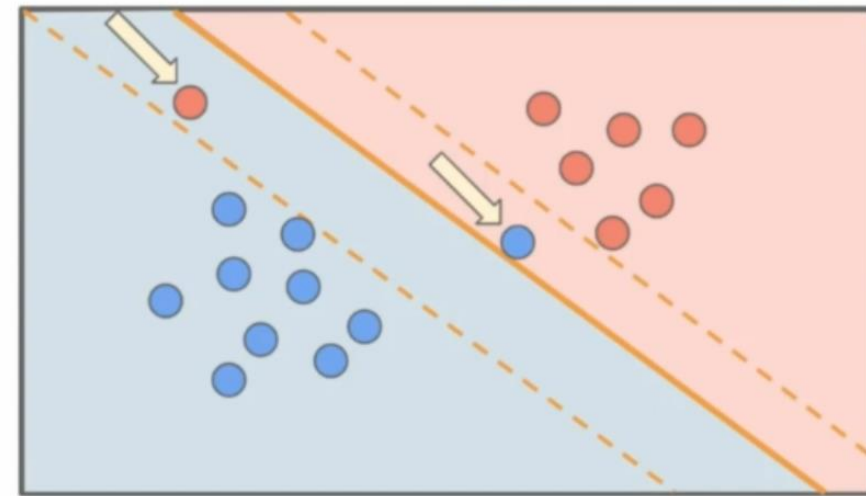


Метод опорных векторов «kernel trick» математика

- Классификатор опорных векторов

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

x2



x1

Support
Vector
Classifier

Метод опорных векторов «kernel trick» математика

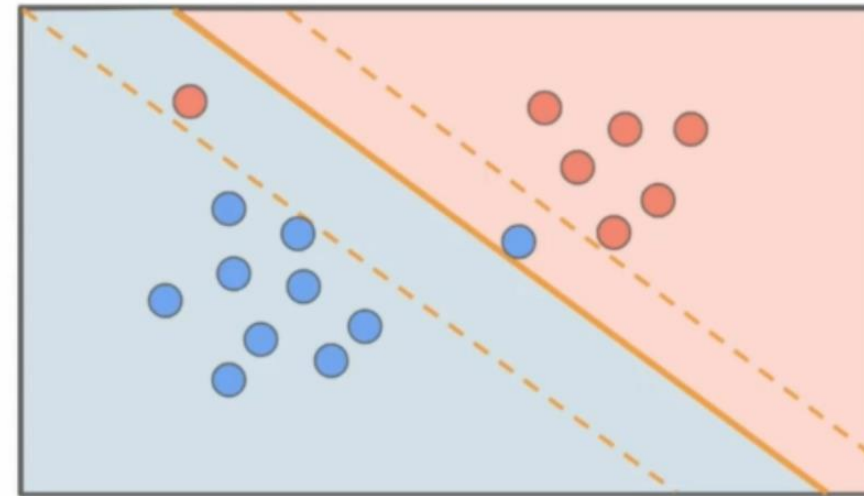
- Классификатор опорных векторов

максимизируем M
 $\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M$

при условии $\sum_{j=1}^p \beta_j^2 = 1$

$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C$

x2



x1

Support
Vector
Classifier

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

Метод опорных векторов «kernel trick» математика

Support
Vector
Classifier

- Классификатор опорных векторов

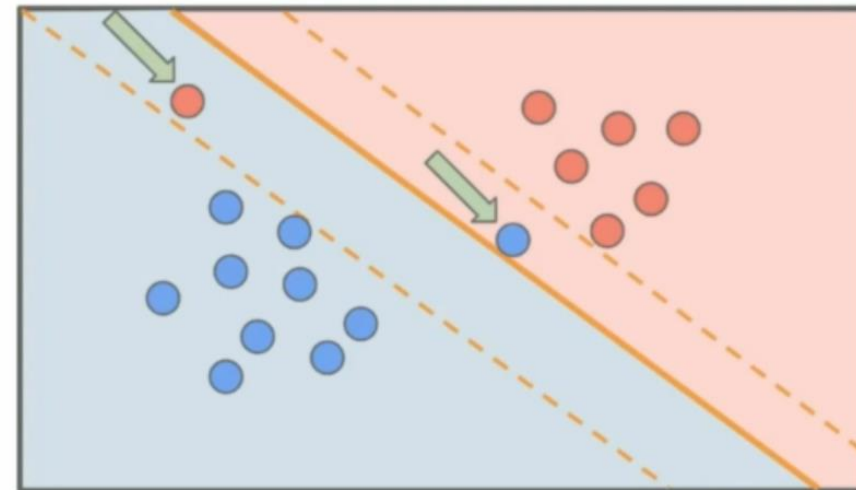
максимизируем M
 $\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M$

при условии $\sum_{j=1}^p \beta_j^2 = 1$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

x2



x1

Метод опорных векторов «kernel trick» математика

- Замечание по поводу SVC в Scikit-Learn

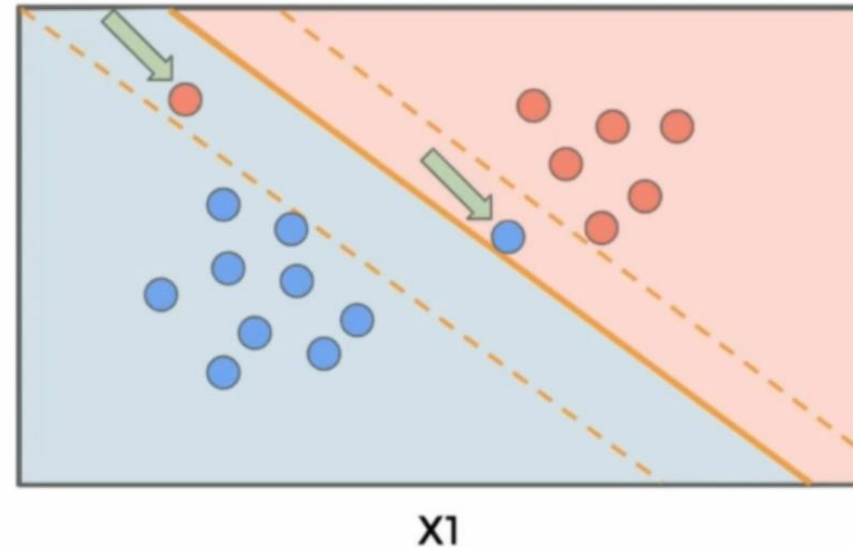
C : float, default=1.0

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

x2

$$\epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$



Чем больше C в формуле, тем больше ошибка

Чем больше C в Scikit-Learn, тем меньше ошибка, благодаря этому можно одинаково трактовать C в разных моделях

Метод опорных векторов «kernel trick» математика

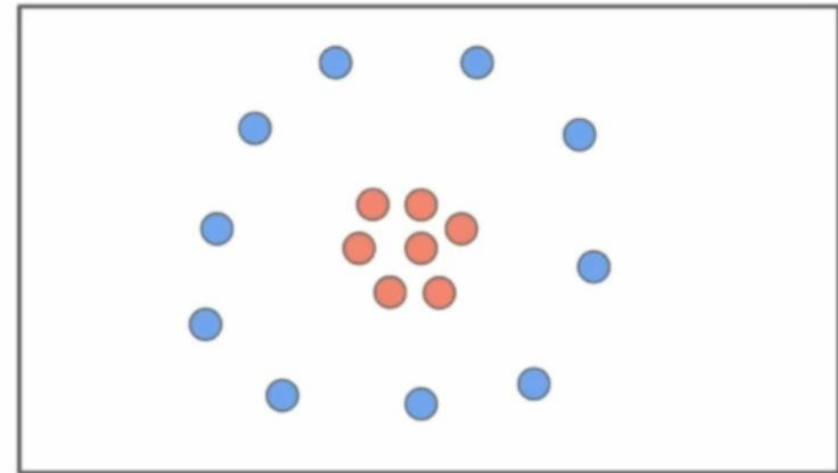
- Метод опорных векторов

$$X_1, X_2, \dots, X_p,$$



$$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$$

x2



x1

Support
Vector
Machines

Метод опорных векторов «kernel trick» математика

- Метод опорных векторов

$$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$$

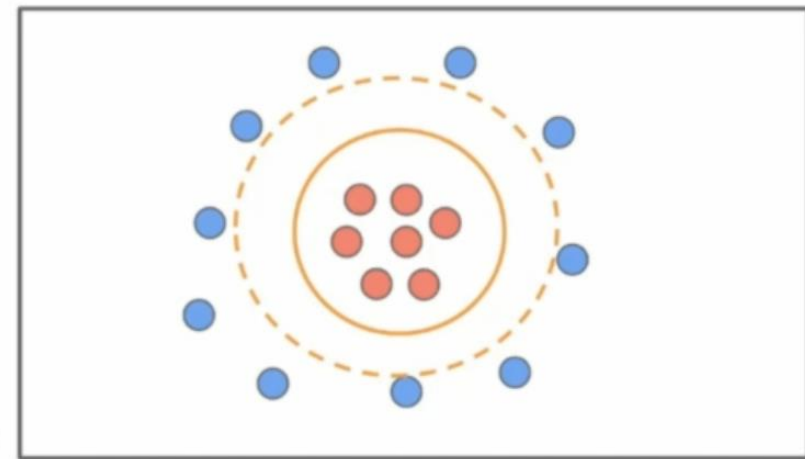
максимизируем M

$$\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n, M$$

при условии $y_i \left(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i)$

$$\sum_{i=1}^n \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1.$$

Support
Vector
Machines



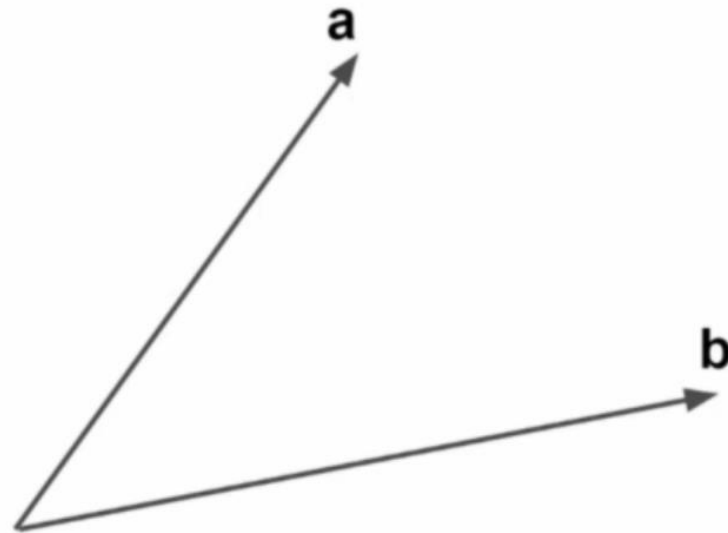
Метод опорных векторов «kernel trick» математика

- Как работать с пространствами очень больших размерностей?
- При росте порядка полиномов растёт и вычислительная трудоёмкость для поиска зазоров.
- Эту сложность помогает решить “kernel trick”, который использует скалярное произведение векторов (dot product)

Метод опорных векторов «kernel trick» математика

- Скалярное произведение векторов (dot product)

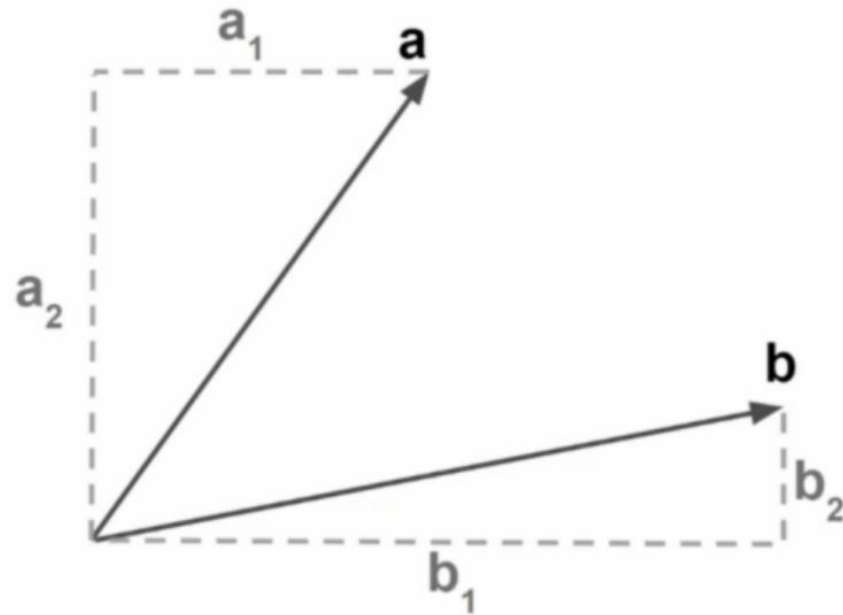
$$\langle a, b \rangle = \sum_{i=1}^r a_i b_i$$



Метод опорных векторов «kernel trick» математика

- Скалярное произведение векторов (dot product)

$$\langle a, b \rangle = \sum_{i=1}^r a_i b_i$$



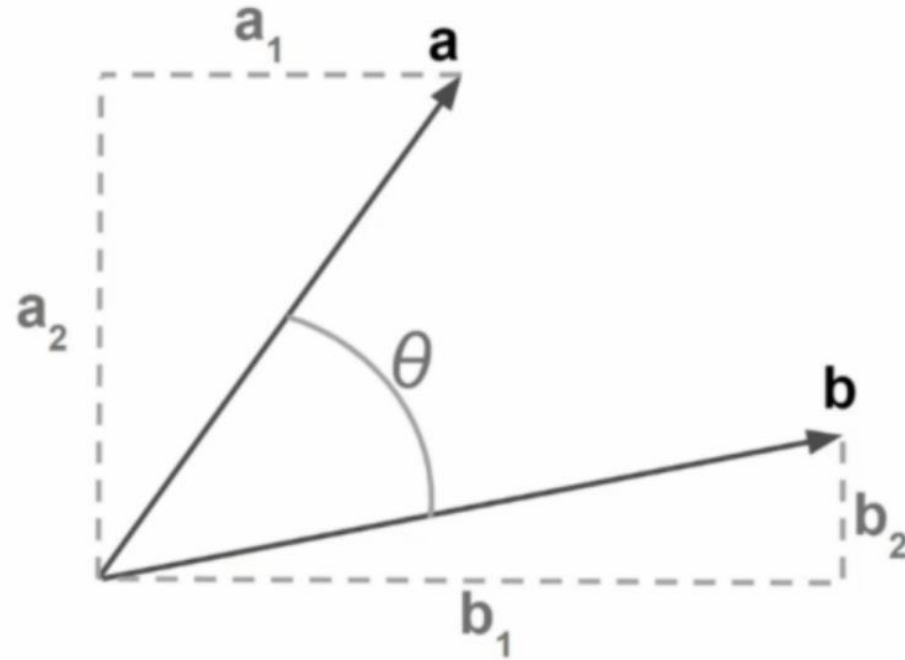
Метод опорных векторов «kernel trick» математика

- Скалярное произведение векторов (dot product)

$$\langle a, b \rangle = \sum_{i=1}^r a_i b_i$$

$$a \cdot b = a_1 b_1 + a_2 b_2$$

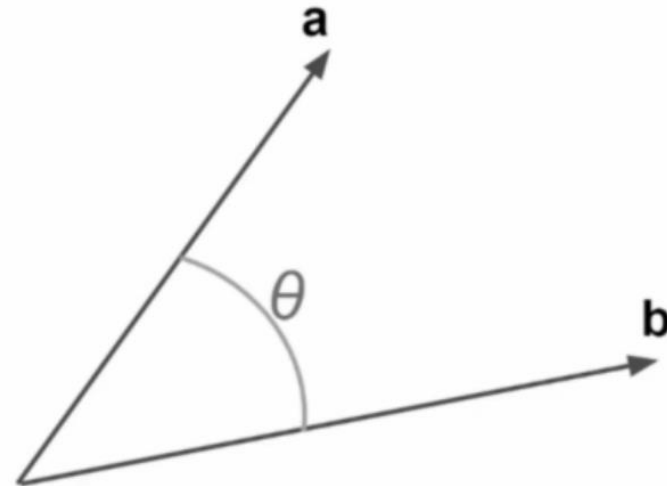
$$a \cdot b = |a||b|\cos(\theta)$$



Метод опорных векторов «kernel trick» математика

- Обратите внимание, что скалярное произведение векторов можно представить себе как “похожесть” этих векторов

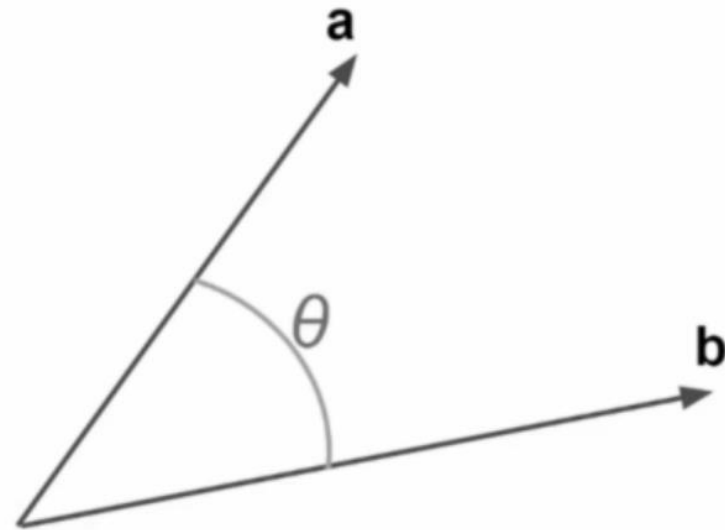
$$a \cdot b = |a||b|\cos(\theta)$$



Метод опорных векторов «kernel trick» математика

- $\cos(0^\circ) = 1$
- $\cos(90^\circ) = 0$
- $\cos(180^\circ) = -1$

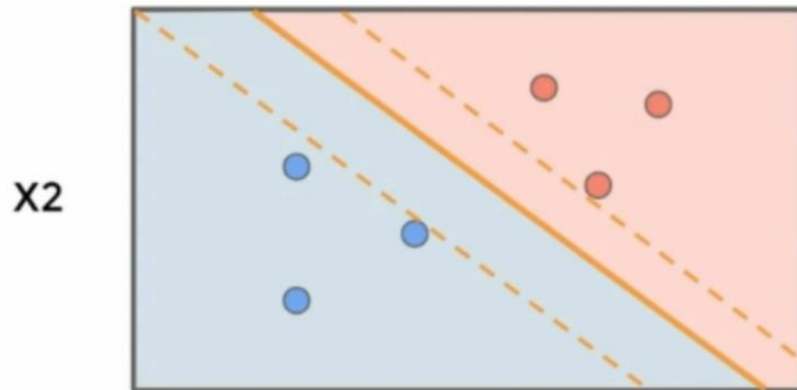
$$a \cdot b = |a||b|\cos(\theta)$$



Метод опорных векторов «kernel trick» математика

- Линейный классификатор можно переписать так:

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

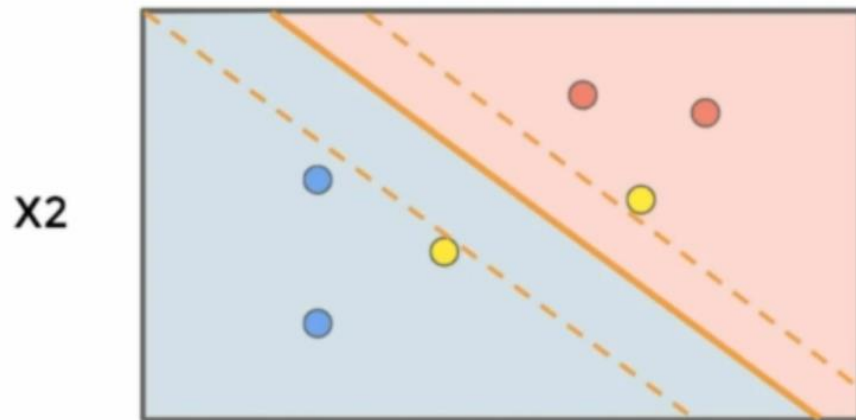


Вычисляем скалярные произведения всех пар обучающих наблюдений

Метод опорных векторов «kernel trick» математика

- Линейный классификатор можно переписать так:

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$



Только не-нули для
опорных векторов

Метод опорных векторов «kernel trick» математика

- Линейный классификатор можно переписать так:

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$



$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$



Можем сократить кол-во рассматриваемых точек до S

Метод опорных векторов «kernel trick» математика

- Функция ядра:

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j}$$

Ядро - функция, численно описывающая похожесть двух наблюдений

Метод опорных векторов «kernel trick» математика

- Функция ядра:

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j} \quad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

Метод опорных векторов «kernel trick» математика

- Функция ядра:

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j} \quad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

The diagram illustrates the relationship between the kernel function K , the function f , and the inner product definition. The kernel function $K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j}$ is shown on the left, and the function $f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$ is shown on the right. Both the summation in K and the inner product term in f are enclosed in red rounded rectangles. Below these, the inner product is defined as $\langle a, b \rangle = \sum_{i=1}^r a_i b_i$, which is enclosed in a red rectangle. Two red curved arrows point from the definition of the inner product to the summation in K and the inner product term in f , indicating that the kernel function and the function f are defined using the inner product.

Метод опорных векторов «kernel trick» математика

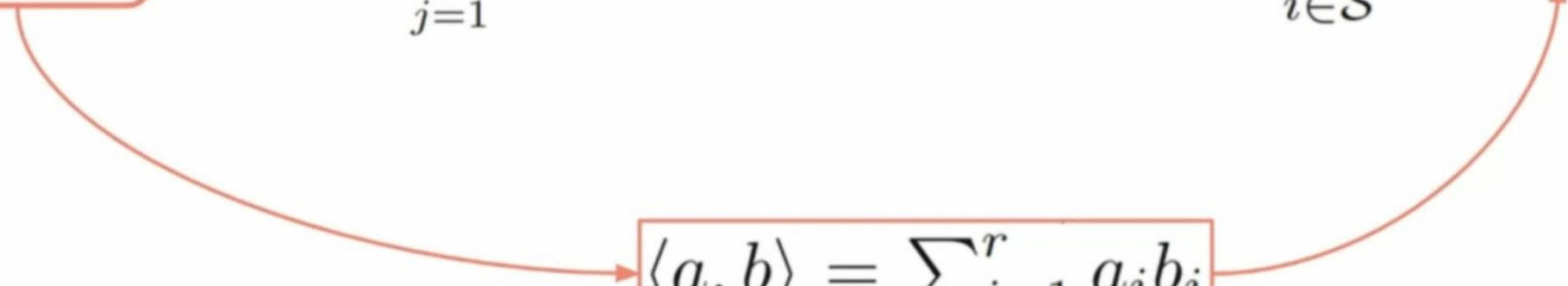
- Полиномиальная функция ядра:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij} x_{i'j}\right)^d \quad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

The diagram illustrates the relationship between the kernel function $K(x_i, x_{i'})$, the inner product $\langle a, b \rangle$, and the polynomial function $f(x)$. Red boxes highlight the expressions $K(x_i, x_{i'})$, $\langle a, b \rangle$, and $\langle x, x_i \rangle$. Red arrows indicate that $K(x_i, x_{i'})$ is a function of $\langle a, b \rangle$, and $\langle x, x_i \rangle$ is a function of $\langle a, b \rangle$.

Метод опорных векторов «kernel trick» математика

- Радиальная базисная функция ядра (radial basis kernel):

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2\right) \quad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

$$\langle a, b \rangle = \sum_{i=1}^r a_i b_i$$

Метод опорных векторов «kernel trick» математика

- Применение функций ядра ещё называют словосочетанием “kernel trick”.
- Ядра позволяют избежать вычислений в увеличенном пространстве признаков, выполняя только вычисления для каждой различной пары обучающих точек

Метод опорных векторов «kernel trick» математика

- Ранее мы видели, что скалярное произведение можно представить себе как меру похожести между векторами.
- Функции ядра можно представить себе как меру похожести между исходным пространством признаков и увеличенным пространством признаков.

Назначение гиперпараметров в SVM

- Параметр C (Cost): Контролирует компромисс между достижением максимальной ширины разделяющей полосы (margin) и минимизацией количества ошибочных классификаций. Большое значение C означает меньшую толерантность к ошибкам (жесткая зазор), в то время как меньшее значение C позволяет больше ошибок (мягкая зазор).
- Параметры ядра: Определяют тип преобразования пространства признаков для преодоления нелинейности. Примеры включают полиномиальное ядро (degree, coef0), радиальное базисное функции (RBF) ядро (gamma), и линейное ядро.
 - Gamma (γ): Только для нелинейных ядер, например RBF. Определяет степень влияния одного тренировочного примера на другие; низкие значения указывают на 'длинный' радиус влияния, высокие значения – на 'короткий'.
 - Degree: Для полиномиального ядра определяет степень полинома.
 - Coef0: Независимый член в полиномиальном или сигмоидальном ядре.

Параметр C в SVM:

Параметр C в методе опорных векторов (SVM) является гиперпараметром регуляризации, который играет ключевую роль в определении компромисса между достижением максимальной ширины разделяющей полосы и минимизацией количества ошибочных классификаций на обучающей выборке. Вот как можно понимать различные значения параметра C в контексте SVM:

Значения параметра C :

- **Малые значения C :**

- **Мягкая зазор:** При малых значениях C модель допускает больше ошибок на обучающих данных. Это создает более "мягкую" маржу, позволяя гиперплоскости лучше обобщаться на новых данных за счет увеличения количества ошибок классификации на обучающей выборке.
- **Меньшее переобучение:** Малые значения C помогают снизить риск переобучения, особенно в случаях, когда количество признаков велико по отношению к количеству образцов.
- **Большая устойчивость:** Модели с меньшим значением C менее чувствительны к выбросам и шуму в данных.

- **Большие значения C :**

- **Жесткая зазор:** Большие значения C стремятся к минимизации количества ошибок на обучающих данных, что приводит к более "жесткой" марже. Это может улучшить точность на обучающей выборке, но также увеличить риск переобучения.
- **Большее переобучение:** Когда C слишком велико, модель может слишком точно подогнать обучающие данные, поймав шум как важные признаки, что ухудшает обобщающую способность.
- **Меньшая устойчивость:** Модели с большим значением C более чувствительны к выбросам, так как они стремятся классифицировать каждый образец правильно.

Параметр kernel в SVM:

- 'linear': Использует линейное ядро. В этом случае пространство признаков остается неизменным, а разделяющая граница между классами является линейной (или гиперплоскостью в многомерном пространстве). Это подходит для данных, которые линейно разделимы или когда количество признаков значительно превышает количество образцов.
- 'poly': Использует полиномиальное ядро. Это позволяет SVM работать в пространстве более высокой размерности, что соответствует использованию полиномиальных комбинаций исходных признаков. Полиномиальное ядро контролируется параметрами степени (degree), коэффициента (coef0) и масштаба.
 - Степень (Degree): Определяет степень полинома. Например, степень 2 соответствует квадратичному ядру, степень 3 – кубическому ядру и так далее.
 - Коэффициент (Coef0): Определяет вклад свободного члена в полиномиальное ядро. Этот параметр может помочь контролировать влияние высших и низших степеней в полиноме.
- 'rbf': Использует радиально-базисную функцию (Radial Basis Function). Это наиболее часто используемое ядро для SVM и подходит для случаев, когда отношения между классами нелинейны. RBF может отображать исходные данные в бесконечномерное пространство, делая возможным эффективное разделение сложнотренируемых данных. Ядро RBF контролируется параметром γ , который определяет степень влияния одного тренировочного примера на другие.
- 'sigmoid': Использует сигмоидное ядро, которое основано на сигмоидной функции, аналогичной используемой в логистической регрессии. Этот тип ядра преобразует исходные пространственные данные в пространство, где они могут быть разделены гиперплоскостью, но используется реже, поскольку может приводить к нелинейным границам решений, которые не всегда соответствуют структуре данных.

SVM для решения задач многоклассовой классификации

- **1. Один против всех (One-vs-All, OvA):**
 - Этот метод заключается в создании отдельного классификатора для каждого класса. Каждый классификатор обучается различать примеры одного класса от всех остальных классов. Например, если у нас есть три класса A, B и C, мы создаем три классификатора:
 - Первый классификатор обучается отличать класс A от классов B и C.
 - Второй классификатор обучается различать класс B от классов A и C.
 - Третий классификатор обучается различать класс C от классов A и B.
 - При классификации нового образца мы используем все классификаторы и выбираем тот класс, для которого соответствующий классификатор выдает наибольшую уверенность или расстояние от разделяющей гиперплоскости.
- **2. Один против одного (One-vs-One, OvO):**
 - В этом подходе создается классификатор для каждой пары классов. Если есть N классов, то будет обучено $N(N-1)/2$ классификаторов. Для трех классов A, B и C будут созданы следующие классификаторы:
 - Классификатор, различающий класс A от класса B.
 - Классификатор, различающий класс A от класса C.
 - Классификатор, различающий класс B от класса C.
 - При классификации нового примера применяются все классификаторы, и класс, который чаще всего выбирается в качестве победителя, присваивается этому примеру.