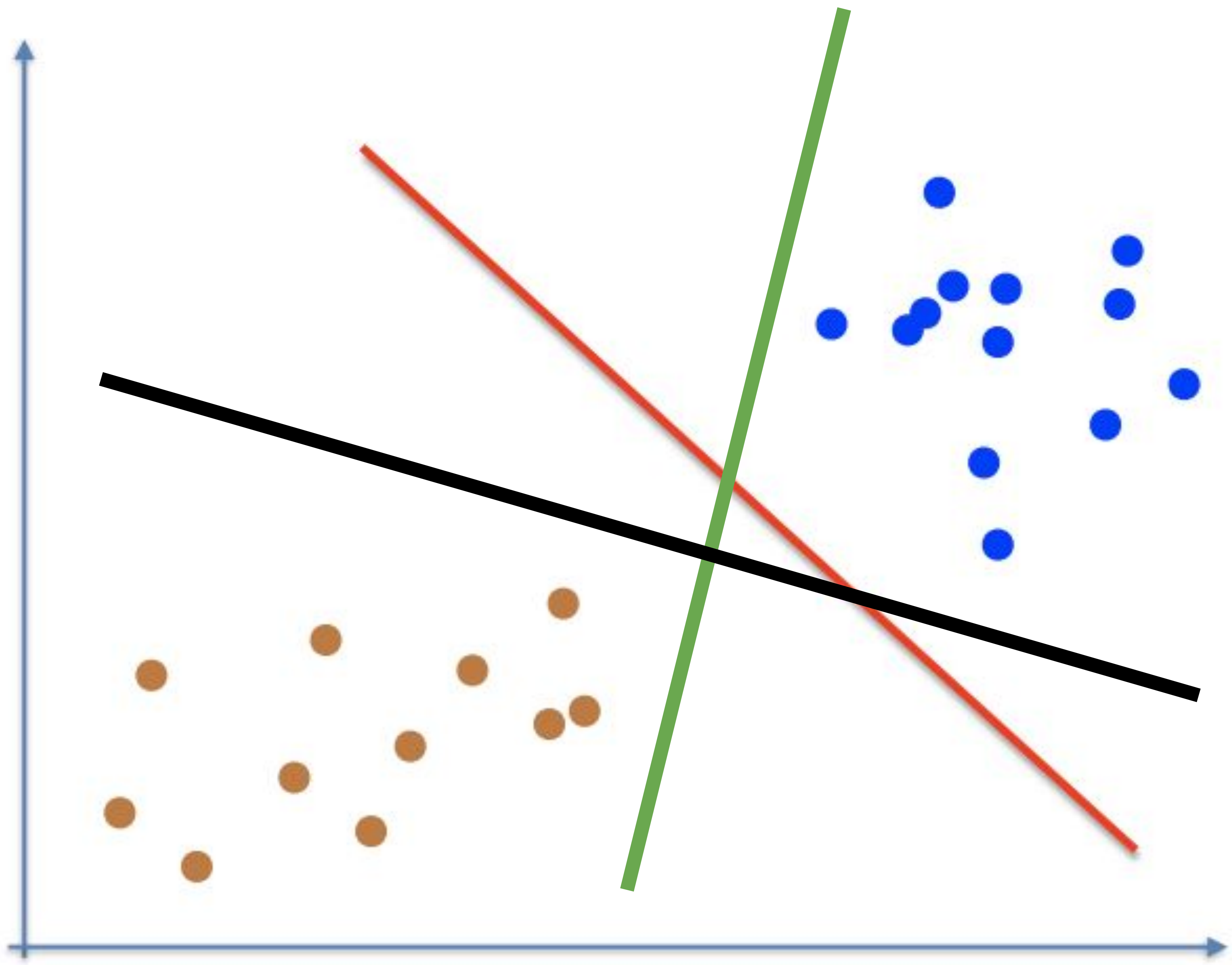
УНИВЕРСИТЕТ
ИННОПОЛИС

SVM. Часть 1

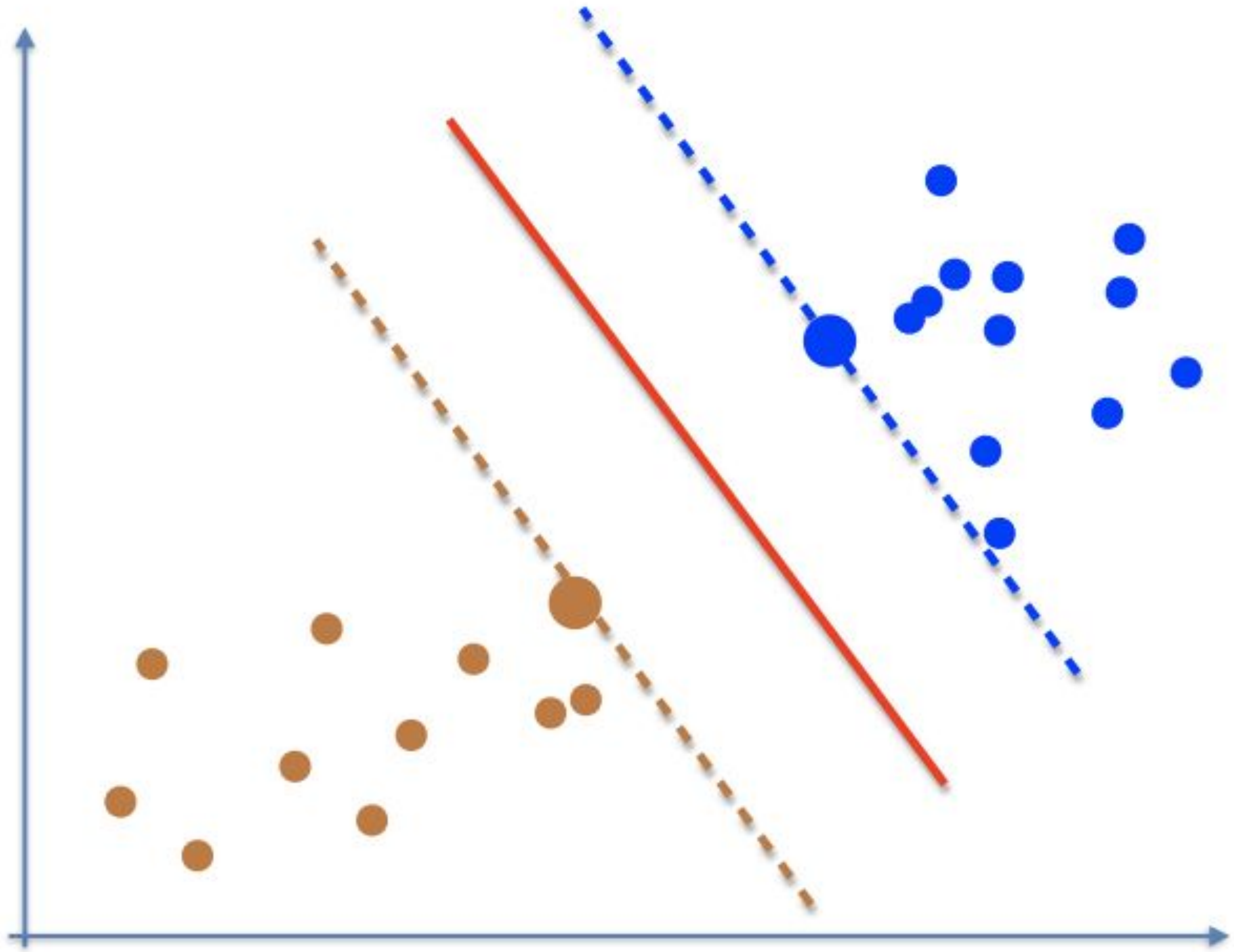
Воробьёва Мария

- maria.vorobyova.ser@gmail.com
- @SparrowMaria

Какая разделяющая прямая лучше?



Какая разделяющая прямая лучше?



SVM подбирает такое уравнение разделяющей гиперплоскости

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_0 = 0$$

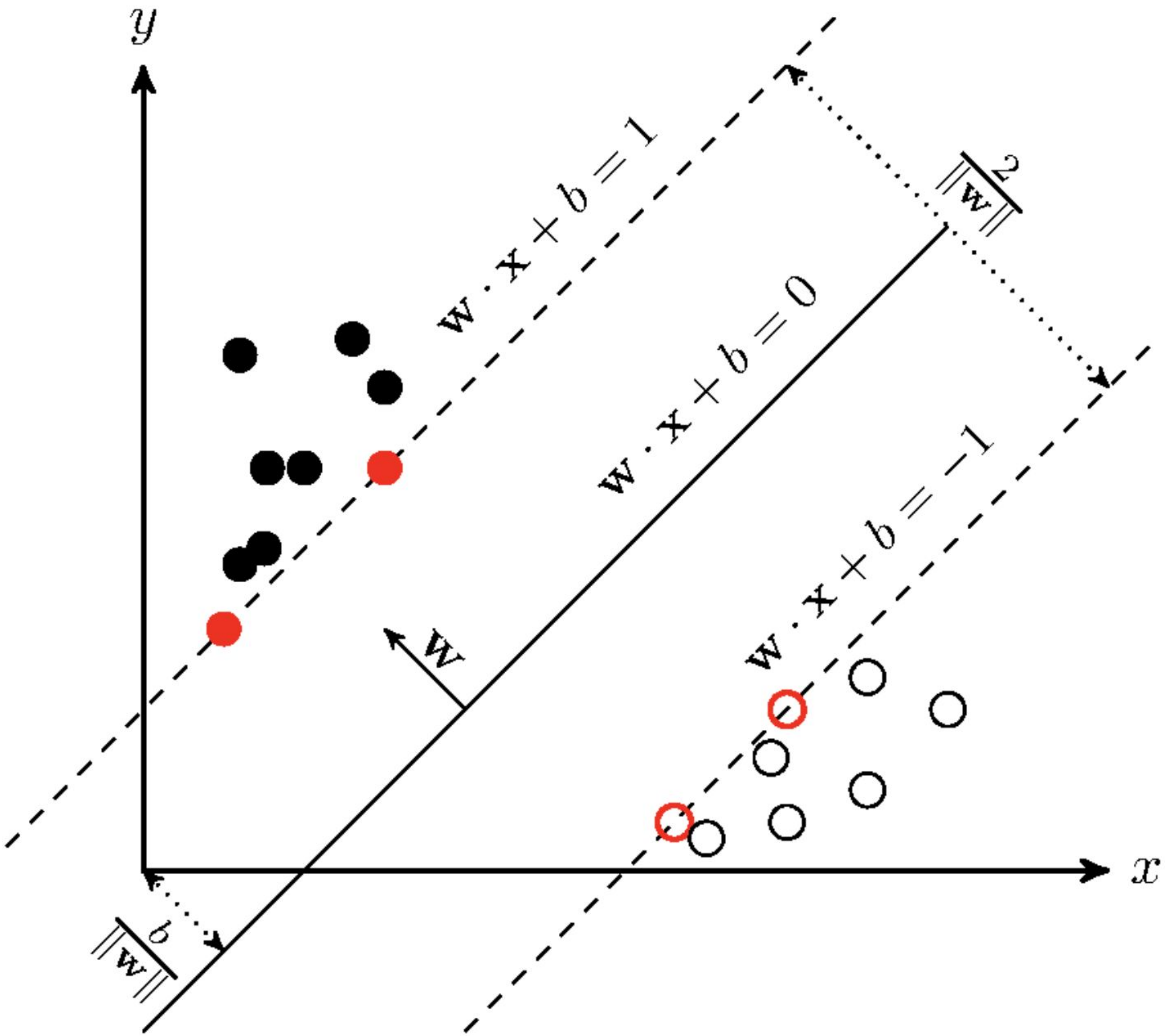
в пространстве R^n, которая бы разделила два класса оптимальным образом

Тогда алгоритм отнесения объекта x к классу Y:

$$F(x) = sign(w^T x - b)$$

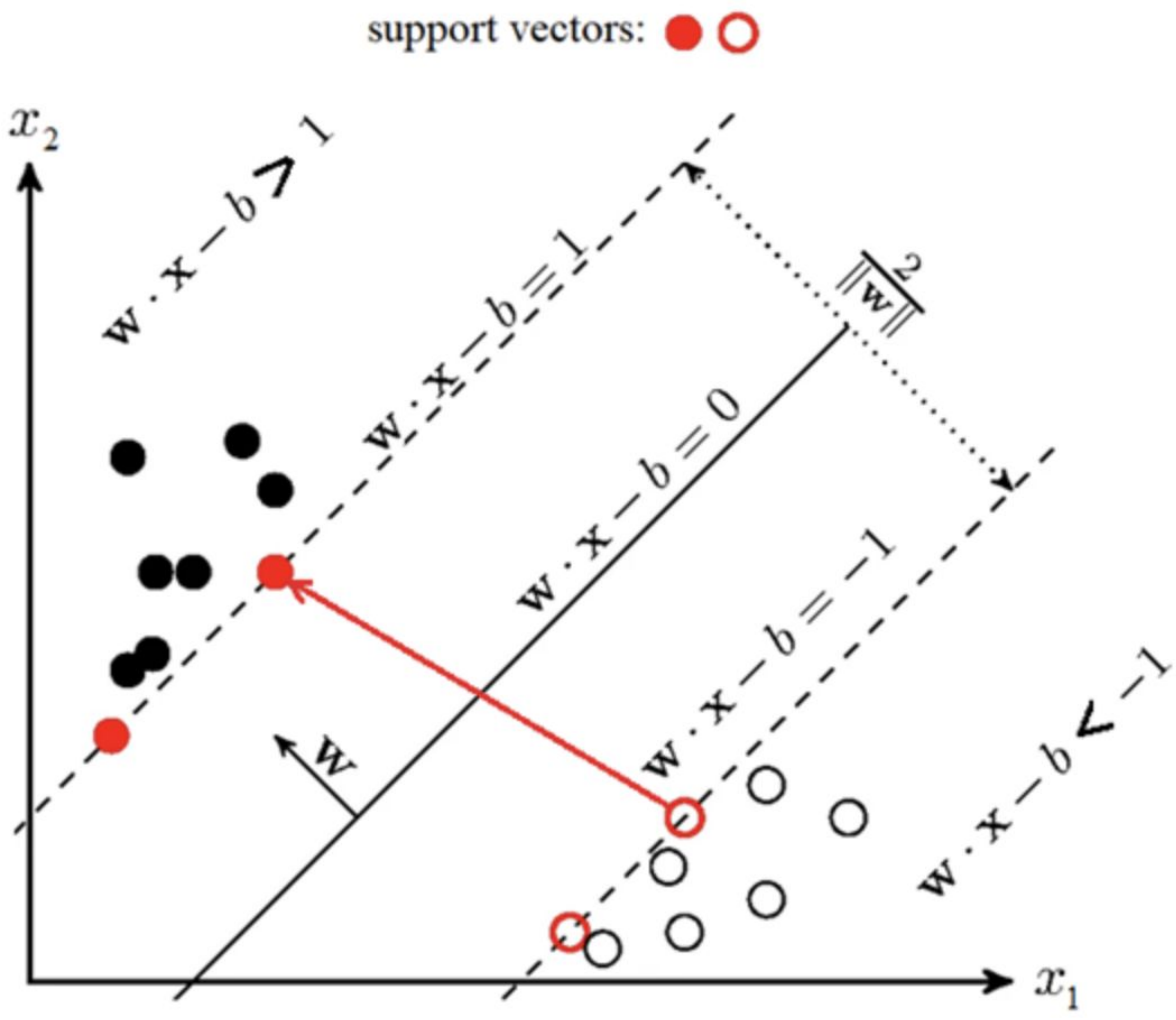
$$w = (w_1, w_2, \ldots, w_n), b = -w_0$$

Алгоритм SVM



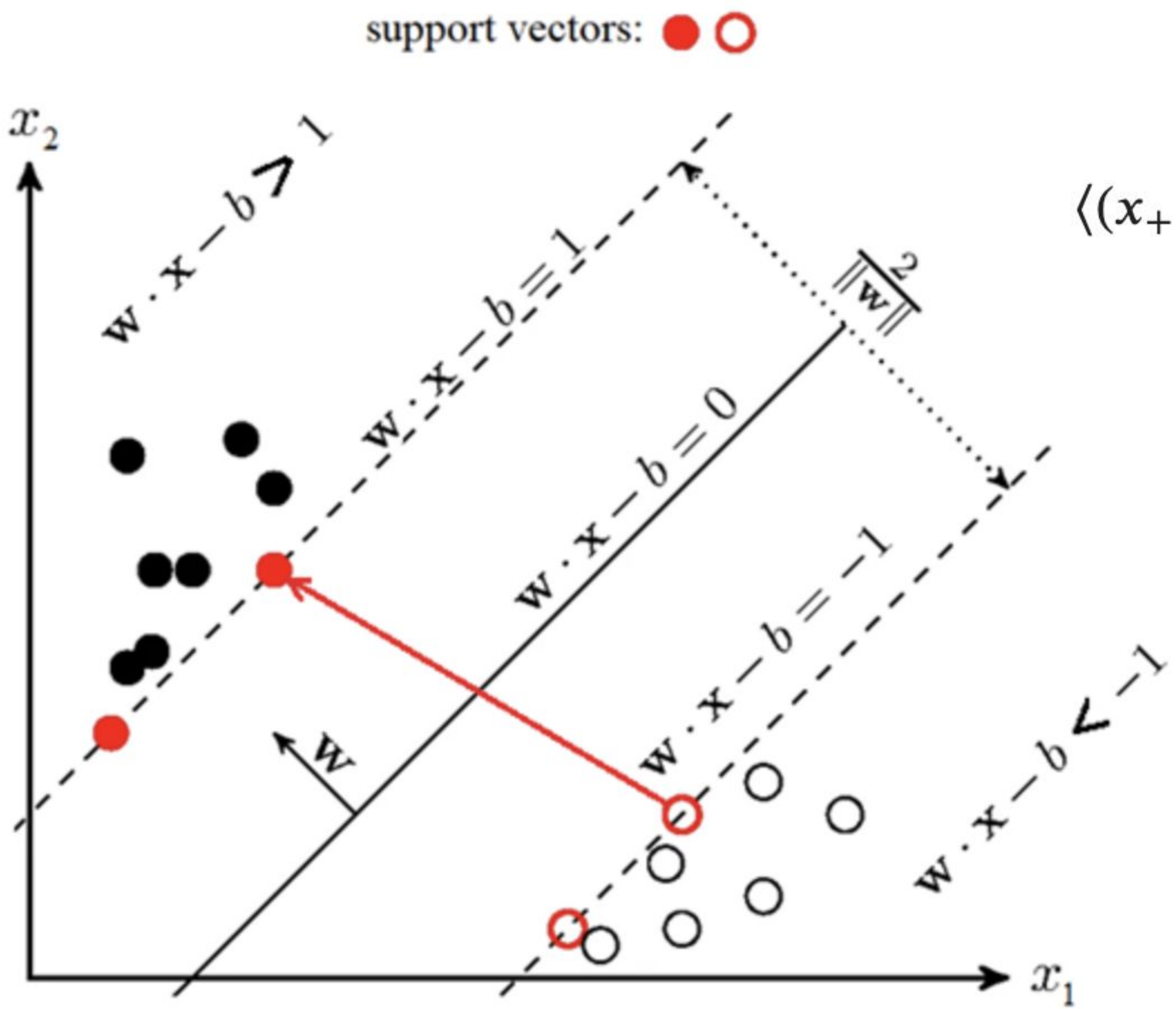
SVM максимизирует зазор (*англ. margin*) между гиперплоскостью и объектами классов, которые расположены ближе всего к ней. Такие объекты и называют опорными векторами .

Алгоритм SVM



1) Пусть вектор *w* — вектор нормали к разделяющей гиперплоскости.
2) далее необходимо найти вектор, концами которого являются опорные вектора
3) далее необходимо найти проекцию вектора из п2 на вектор w
4) проекция и будет показывать ширину разделяющий полосы

Алгоритм SVM

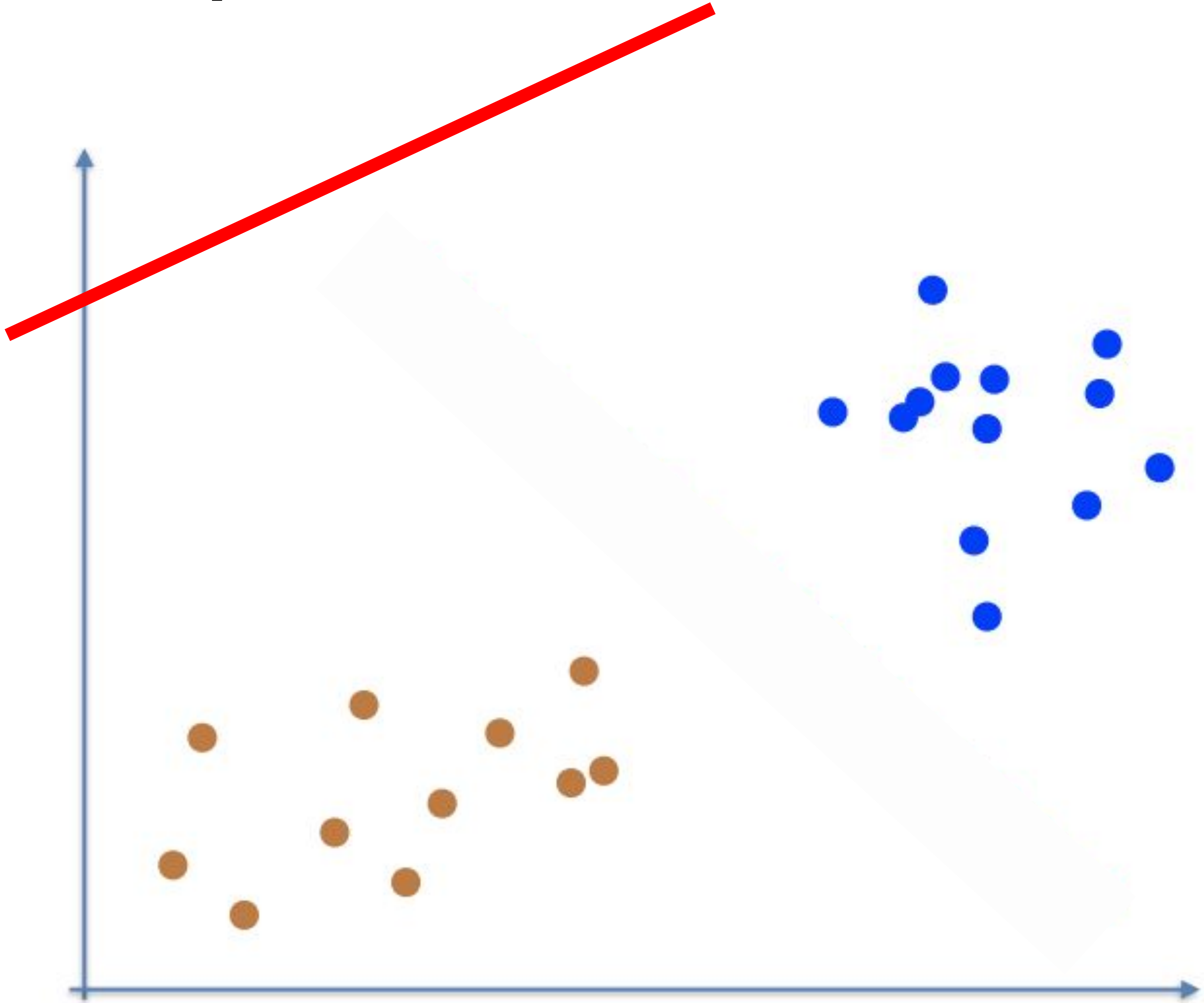


$$\left\langle (x_+ - x_-), \frac{w}{\|w\|} \right\rangle = \frac{\langle x_+, w\rangle - \langle x_-, w\rangle}{\|w\|} = \frac{(b+1)-(b-1)}{\|w\|} = \frac{2}{\|w\|}$$

$$\frac{2}{\|w\|} \rightarrow max \qquad \frac{\|w\|}{2} \rightarrow min$$

$$\frac{w^T w}{2} \rightarrow min$$

Алгоритм SVM

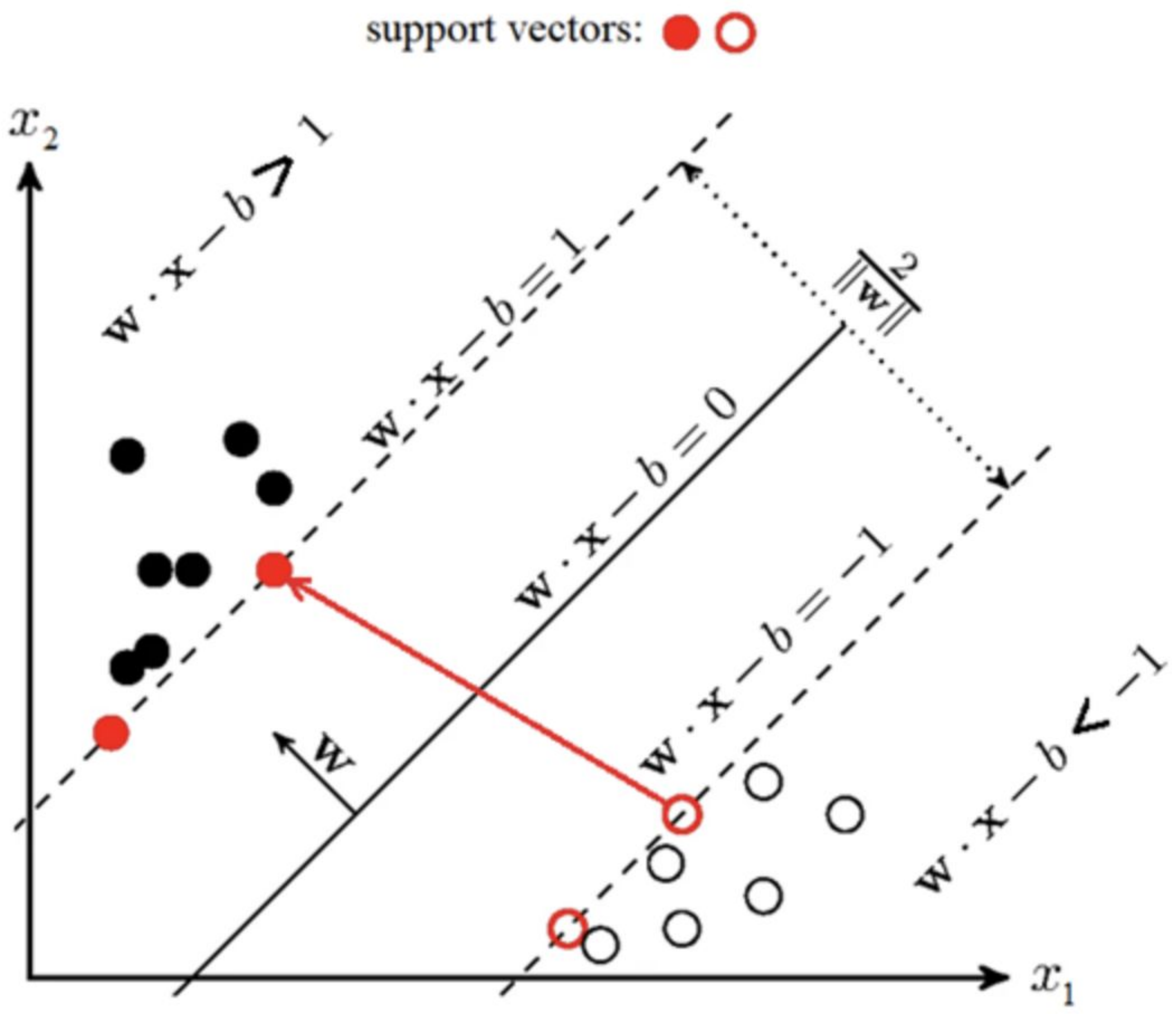


Если будем ориентироваться только на это ограничение,

$$\frac{w^T w}{2} \rightarrow min$$

то можем получить например, вот такой результат

Алгоритм SVM



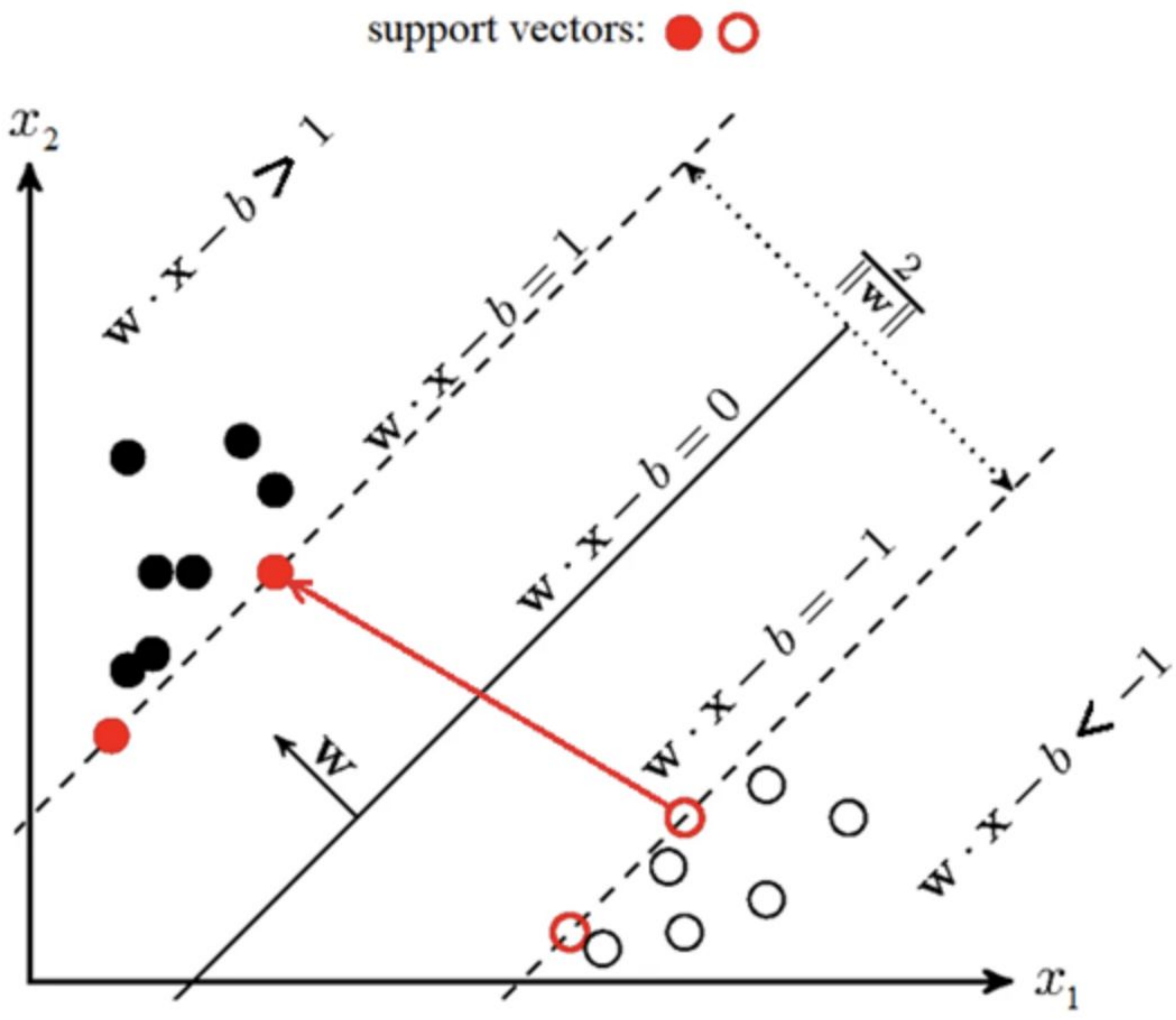
Отступом (*англ. margin*) объекта x от границы классов называется величина

$$M = y(w^T x - b)$$

Алгоритм допускает ошибку на объекте тогда и только тогда, когда отступ отрицателен, то есть если M>1, то объект классифицируется правильно

$$y(w^T x - b) \geqslant 1$$

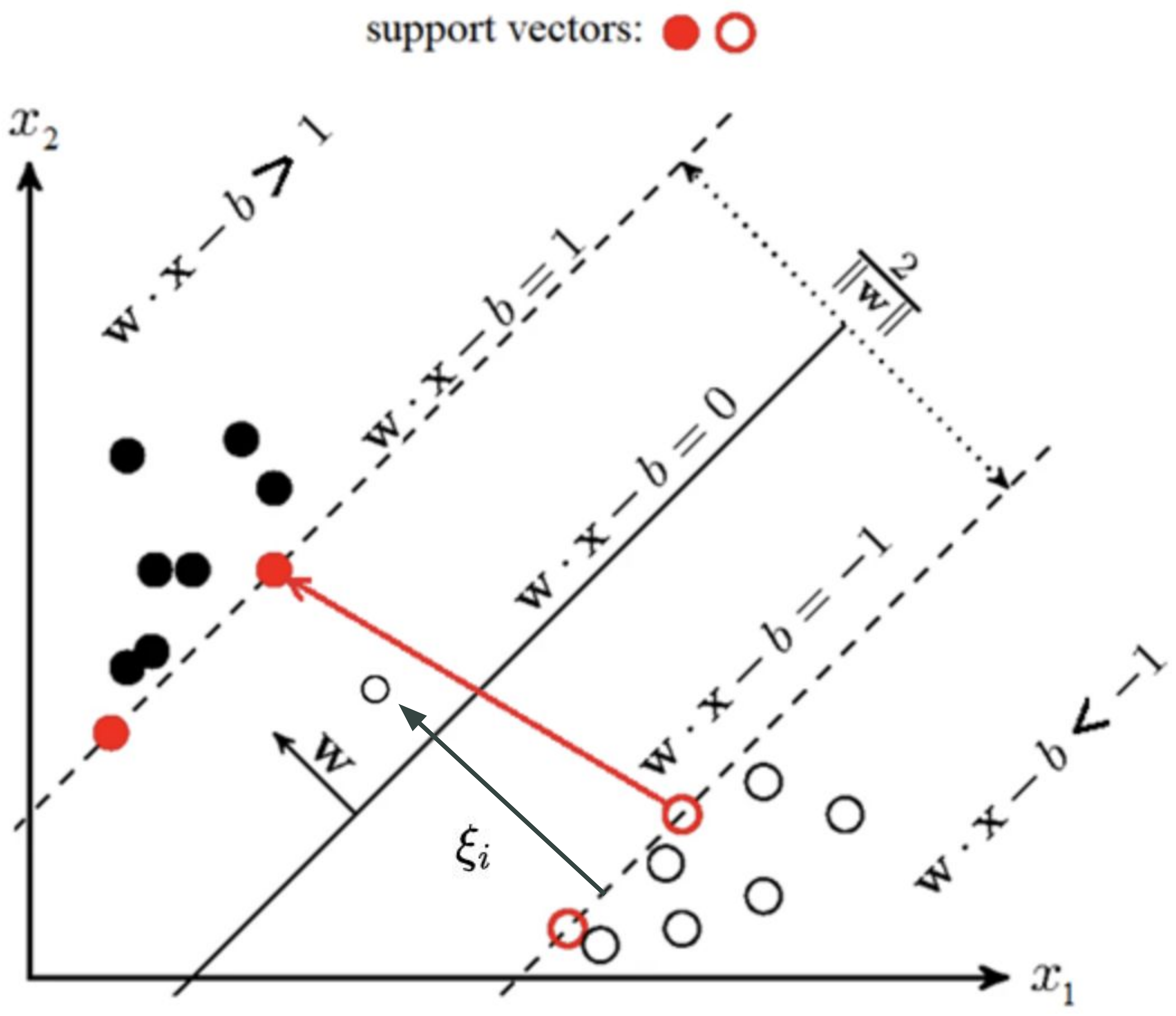
Алгоритм SVM



В итоге мы получаем дефолтную настройку SVM с жестким зазором (hard-margin SVM)

$$\begin{cases} (w^T w)/2 \to min \\ y(w^T x - b) \geqslant 1 \end{cases}$$

Алгоритм SVM



Пусть алгоритм допускает ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было поменьше.

$\xi_i > 0$ - величина ошибки на каждом объекте x_i

$$\begin{cases} (w^T w)/2 + \alpha \sum \xi_i \to min \\ y(w^T x_i - b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

Алгоритм SVM

Будем считать количество ошибок алгоритма (когда M<0). Назовем данную сумму штрафом (Penalty). Тогда штраф для всех объектов будет равен сумме штрафов для каждого объекта x_i , где $[M_i < 0]$ пороговая функция

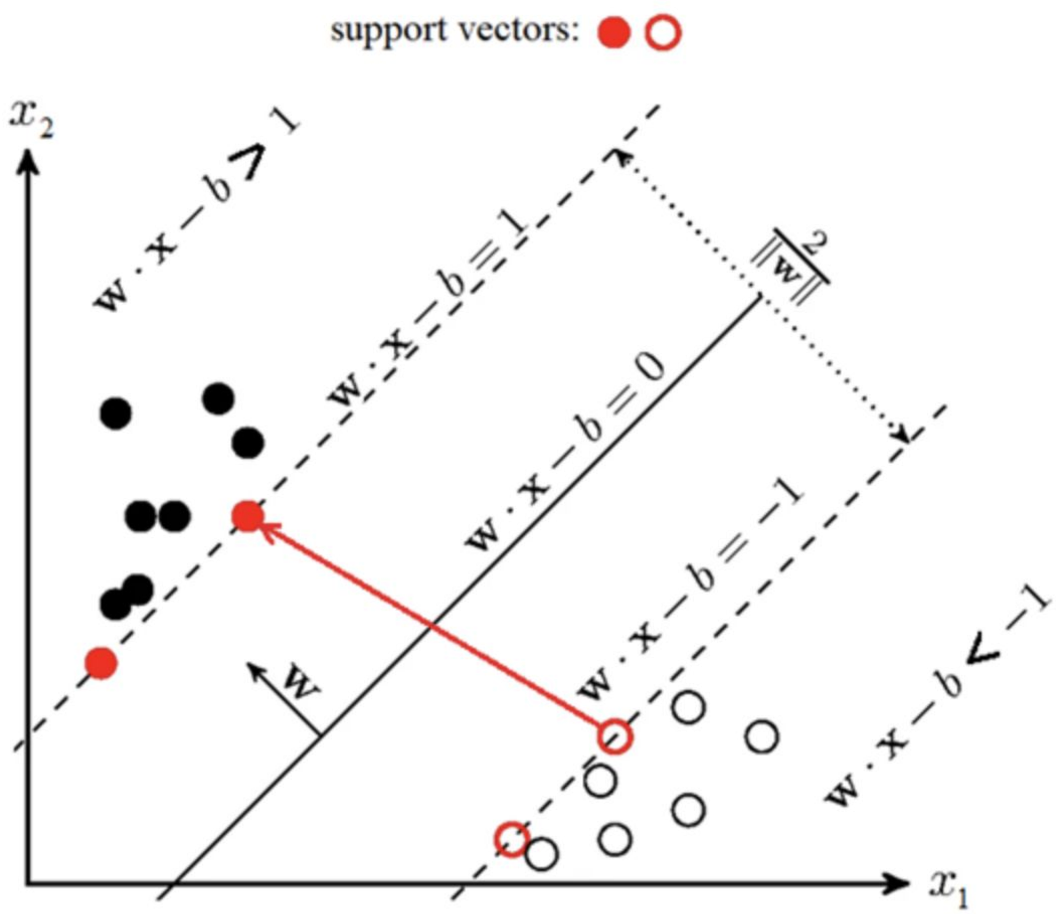
$$Penalty = \sum [M_i < 0]$$

$$[M_i < 0] = \begin{cases} 1 & \text{, если } M_i < 0 \\ 0 & \text{, если } M_i \geqslant 0 \end{cases}$$

Алгоритм SVM

Далее сделаем штраф чувствительным к величине ошибки (чем сильнее "уходит в минус" — тем больше штраф) и заодно введем штраф за приближение объекта к границе классов.

Для этого возьмем функцию, которая ограничивает пороговую функцию ошибки



$$Penalty = \sum [M_i < 0] \leqslant \sum (1 - M_i)_+ = \sum max(0, 1 - M_i)$$

Алгоритм SVM

При добавлении к выражению штрафа слагаемое $\alpha(w^T w)/2$ получаем классическую фукцию потерь SVM с мягким зазором (soft-margin SVM) для одного объекта:

$$Q = max(0, 1 - M_i) + \alpha(w^T w)/2$$

$$Q = max(0, 1 - yw^T x) + \alpha(w^T w)/2$$

Q — функция потерь, она же loss function. Именно ее мы и будем минимизировать с помощью градиентного спуска

$$Q(X, w) = C \sum_{i=1}^{l} \max\{0, 1 - y_i(\langle x_i, w \rangle)\} + \|w\|^2 \to \min_w$$

доп материал https://habr.com/ru/companies/ods/articles/484148/

Алгоритм SVM. Плюсы

1) Хорошо работает с пространством признаков большого размера
   SVM эффективен в задачах, где количество признаков превышает количество образцов, что делает его подходящим для высокоразмерных данных.

2) Хорошо работает с данными небольшого объема:
   При малом количестве обучающих примеров SVM показывает хорошие результаты благодаря своей способности находить оптимальную разделяющую гиперплоскость.

3) Максимизация разделяющей полосы:
   SVM стремится максимизировать полосу, разделяющую классы, что снижает вероятность ошибок классификации и улучшает обобщающую способность модели

4) Единственное решение задачи:
   Решение задачи SVM сводится к задаче квадратичного программирования в выпуклой области, что гарантирует существование единственного решения и стабильность модели.

Алгоритм SVM. Минусы

1) Долгое время обучения:

Для больших наборов данных обучение SVM может занимать много времени и требовать значительных вычислительных ресурсов.

2) Неустойчивость к шуму:

SVM чувствителен к выбросам и шуму в данных, так как выбросы могут стать опорными объектами, что негативно влияет на построение разделяющей гиперплоскости.

3) Сложности в выборе ядра и спрямляющего пространства:

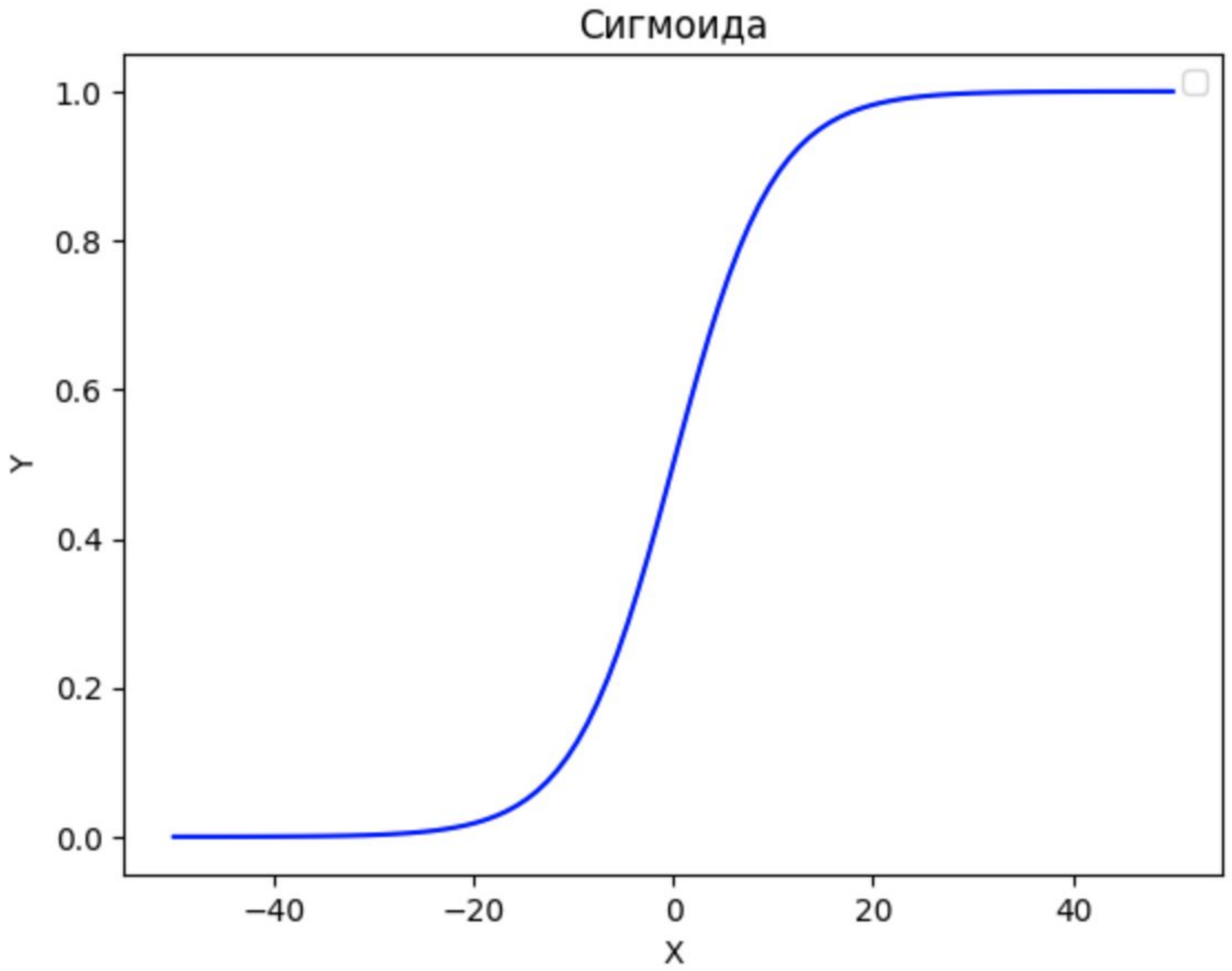
Нет общепринятых методов для выбора наиболее подходящего ядра и спрямляющего пространства для конкретной задачи. Подбор параметров и преобразований данных часто требует экспертных знаний и интуиции.

Алгоритм SVM. Применение SVM

1) задачи с небольшим набором данных;
2) задачи текстовой классификации.

   SVM дает неплохой baseline ([preprocessing] + [TF-iDF] + [SVM]), получаемая точность прогноза оказывается на уровне некоторых сверточных/рекуррентных нейронных сетей
3) для многих задач со структурированными данными связка [feature engineering] + [SVM] + [kernel] работает ;

Логистическая регрессия. Напоминание



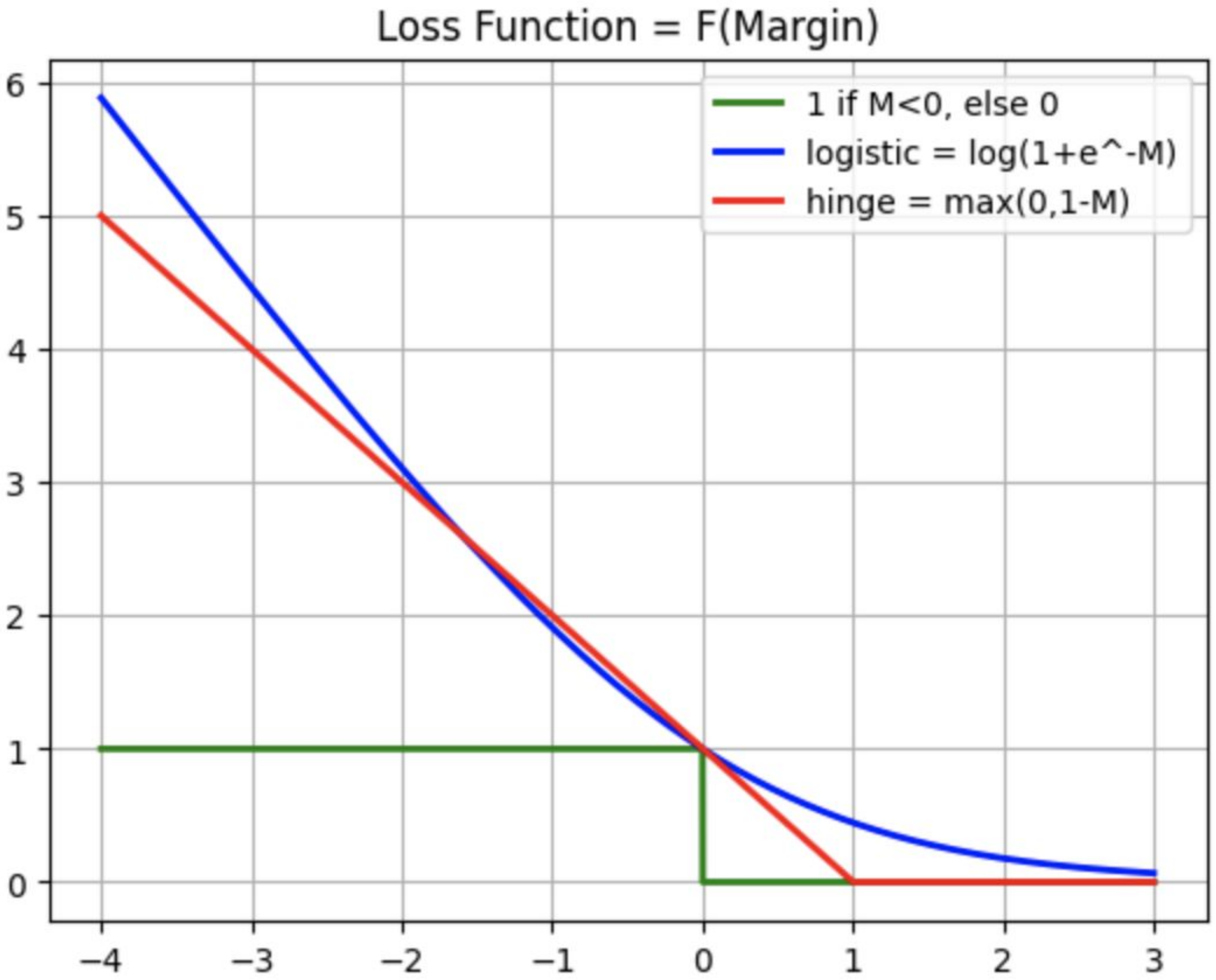
$$\sigma(\langle w, x \rangle) = \frac{1}{1 + \exp(-\langle w, x \rangle)}$$

Если ответы 0 и 1, то

$$\ell(w, X, y) = \sum_i \left(y_i \log(\sigma(\langle w, x_i \rangle)) + (1 - y_i) \log(\sigma(-\langle w, x_i \rangle))\right)$$

Если ответы -1 и 1, то Loss Function выглядит компактнее $log(1 + exp(-y(w^T x)))$

Алгоритм SVM



Hinge loss

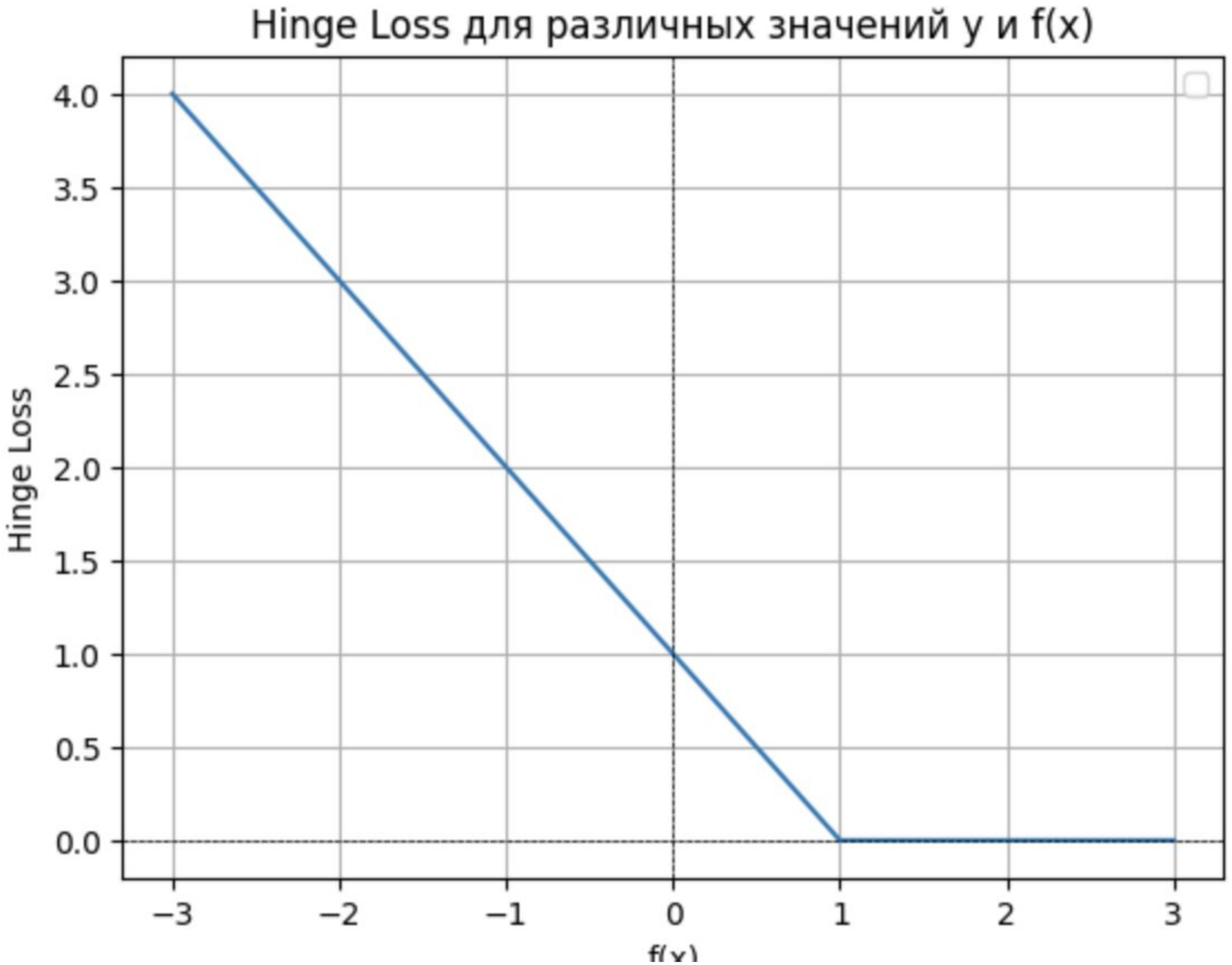
1) относительно устойчив к выбросам в данных.(если количество выбросов мало)

1) Классифицирует строго -1/1 (или 0/1)

Logistic:

1) Гладкая функция, что упрощает оптимизацию
2) Вероятностная интерпретация: модель, обученная с использованием logistic loss, может выдавать вероятности принадлежности к классу

Алгоритм SVM. Hinge Loss



$$Q(X, w) = C \sum_{i=1}^{l} \underbrace{\max\{0, 1 - y_i(\langle x_i, w \rangle)\}}_{\text{Hinge loss для одного объекта}} + \|w\|^2 \to \min_w$$

Hinge loss штрафует неправильные предсказания, пропорционально их отступу от правильного класса.

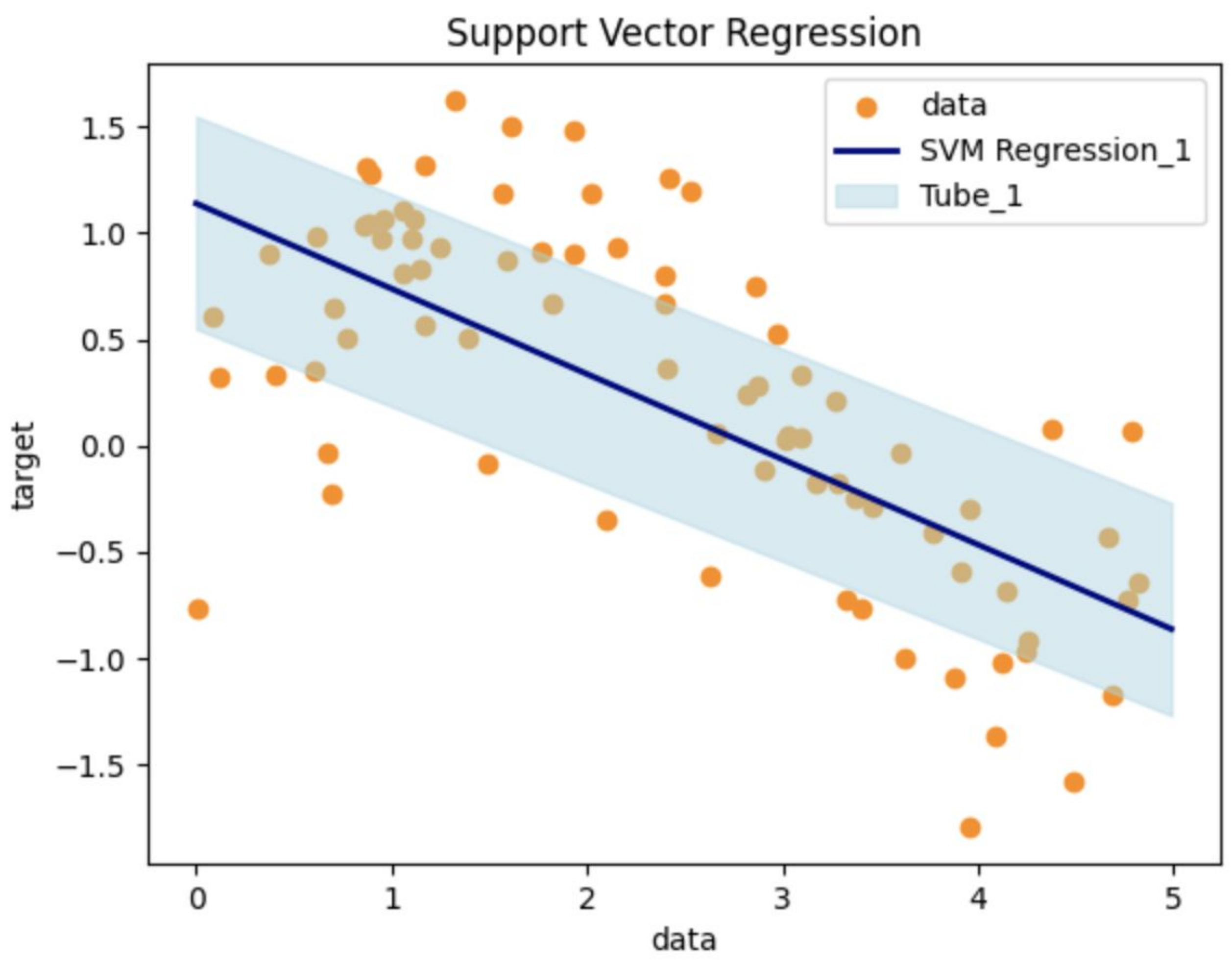
Большое значение C:

- hinge loss -> меньше
- точность на обучающей выборке -> больше ->риск переобучения

Маленькое значение C

- hinge loss -> больше
- точность на обучающей выборке -> меньше ->риск недообучения

Алгоритм SVM. Регрессия



Функция потерь (loss function) SVR выглядит следующим образом:

L(y, f(x)) = max(0, |y - f(x)| - epsilon)

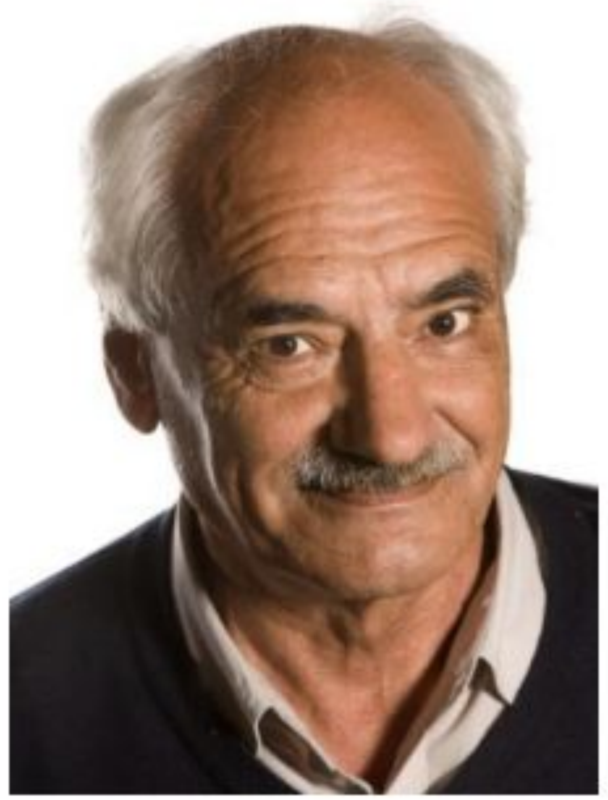
где:

- y - истинное целевое значение для примера x
- f(x) - предсказанное значение модели для примера x
- epsilon - параметр, который определяет интервал, в котором модель считается несущественно ошибающейся.

Для случая, когда |y - f(x)| <= epsilon, функция потерь равна 0, так как ошибка модели находится внутри допустимого интервала. Если |y - f(x)| > epsilon, то модель штрафуется за ошибку, и функция потерь возрастает линейно.

Алгоритм SVM

Вапник Владимир Наумович



Алексей Яковлевич Червоненкис

Метод опорных векторов (support vector), называемый ранее алгоритмом “обобщенного портрета”, был разработан советскими математиками **В. Н. Вапником** и **А. Я. Червоненкисом** в 1974 и с тех пор приобрел широкую популярность

УНИВЕРСИТЕТ
ИННОПОЛИС

ВОПРОСЫ И ОТВЕТЫ