«Машинное обучение»

Линейные методы:
Часть 2. – Линейная классификация

Александр Дьяконов

18 октября 2021 года

План

Линейные решающие модели в задаче бинарной классификации

Идея максимального зазора

Метод опорных векторов (SVM)

Линейные решающие модели в задаче бинарной классификации

Пусть $X = \mathbb{R}^n, Y = \{\pm 1\}$

(обратите внимание на метки)

обучающая выборка: $\{(x_i, y_i)\}_{i=1}^m$

хотим линейную модель:

(формально зависимость нелинейная)

$$a(x) = \operatorname{sgn}(w^{\mathrm{T}} x + b) = \begin{cases} +1, & w^{\mathrm{T}} x + b > 0 \\ -1, & w^{\mathrm{T}} x + b < 0 \end{cases}$$

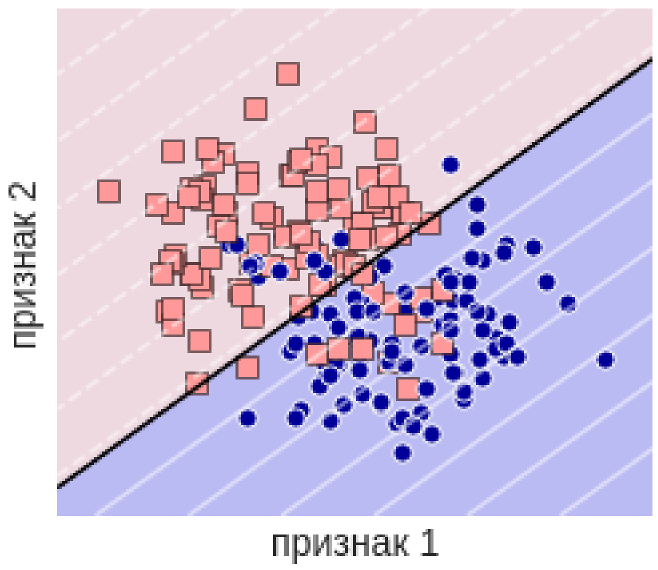
случай $w^{\mathrm{T}} x + b = 0$ **нам тут не особо важен**

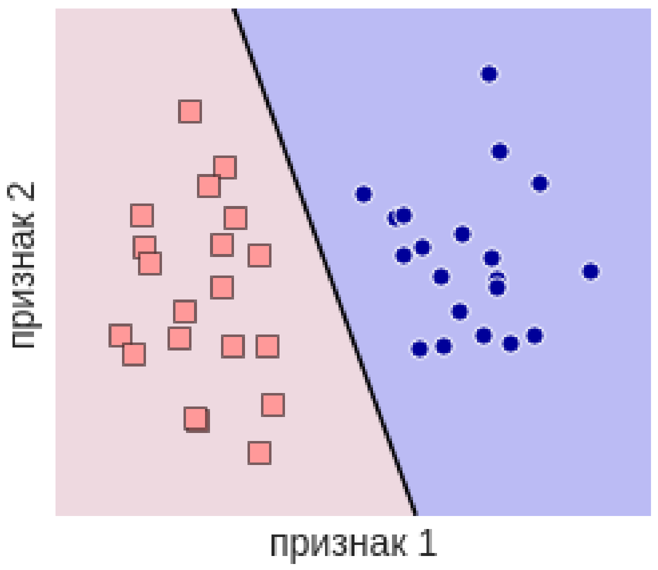
Линейный классификатор

$$a(x) = \operatorname{sgn}(w^{\mathrm{T}} x)$$

если пополнить признаковое пространство фиктивным признаком

Геометрический смысл: делим пространство гиперплоскостью на две части





Иногда это может здорово работать!

Линейный классификатор: обучение

Общая идея – 0-1-loss – число ошибок

$$L(X_{\text{train}},a)=\sum_{t=1}^{m}L(y_t,a(x_t))\to\min$$

$$L(y_t,a(x_t))=\begin{cases}1, & \operatorname{sgn} w^{\mathrm{T}}x_t\neq y_t\\ 0, & \operatorname{sgn} w^{\mathrm{T}}x_t=y_t,\end{cases}$$

или

$$L(y_t,a(x_t))=\theta(-y_t w^{\mathrm{T}}x_t)$$

где $\theta(z)=I[z>0]$

естественно минимизировать число ошибок,

но

- **функция не дифференцируема**
- **выдаёт мало информации**
  только число ошибок, а не их «фатальность»
- **оптимизация здесь – NP-полная задача**

зазор (margin) – $z_t=y_t w^{\mathrm{T}}x_t$ **(чем меньше, тем хуже)**

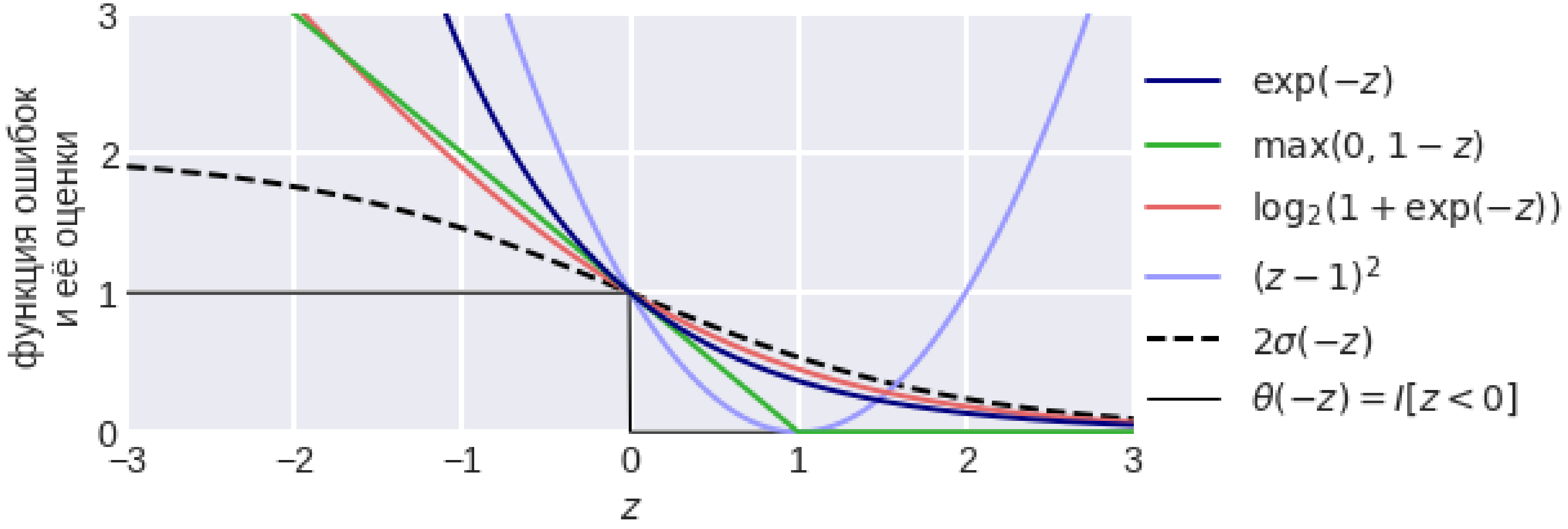
пропорционален расстоянию до ГП с точностью до $\|w\|$ **(дальше), знак ~ ошибка**

Суррогатные функции (surrogate loss functions)

$$\sum_{t=1}^{m} L(y_t, a(x_t)) \le \sum_{t=1}^{m} L'(y_t, a(x_t)) \to \min$$

$$L(y_t, a(x_t)) = \theta(-z_t) \le f(-z_t) = L'(y_t, a(x_t))$$

$$z_t = y_t w^{\mathrm{T}} x_t$$



оценка функции ошибок через гладкую функцию, которую проще оптимизировать

Оценка функции ошибок через гладкую функцию: примеры замен

$$f(-z) = \exp(-z)$$

$$f(-z) = \max(0, 1 - z)$$

$$f(-z) = \log_2(1 + \exp(-z))$$

Обучение – минимизация оценки на обучающей выборке (+ регуляризация)

Персептрон, SVM, логистическая регрессия минимизируют выпуклые аппроксимации 0-1-loss, подменяя NP-трудную задачу задачей выпуклой оптимизации

Пример персептронного алгоритма

$$f(-z) = \max(0, -z)$$

Персептрон: $$\sum_{i=1}^{m} \max[0, -y_i(w^{\mathrm{T}} x_i)] \to \min$$

SGD в персептроне: $$w \leftarrow w + \eta \begin{cases} 0, & \operatorname{sgn}(w^{\mathrm{T}} x_i) = y_i, \\ +x_i & w^{\mathrm{T}} x_i \le 0, y_i = +1, \\ -x_i, & w^{\mathrm{T}} x_i \ge 0, y_i = -1, \end{cases}$$

Почему работает…

$$w^{\mathrm{T}} x_i \le 0, y_i = +1 \Rightarrow w \leftarrow w + \eta x_i \Rightarrow w^{\mathrm{T}} x_i \leftarrow w^{\mathrm{T}} x_i + \eta \underbrace{x_i^{\mathrm{T}} x_i}_{\|x_i\|_2^2 > 0}$$

Пример персептронного алгоритма

Есть теорема Новикова

Если две конечные выборки линейно разделимы,
то разделяющая поверхность находится персептронным алгоритмом
за конечное число шагов
есть и другие способы разделения…

Сравним:

персептронный алгоритм

$$w \leftarrow w + \eta I[-y_i w^{\mathrm{T}} x_i \geq 0] y_i x_i$$

логистическая регрессия (вспомним)

$$w \leftarrow w + \eta \sigma(-y_i w^{\mathrm{T}} x_i) y_i x_i$$

Пример персептронного алгоритма – решение системы линейных неравенств

$$\begin{cases} 2w_1 + w_2 > 0 \\ -w_1 > 0 \\ w_1 - w_2 < 0 \\ -2w_1 - 2w_2 < 0 \end{cases}$$

$$\begin{bmatrix} 2 & 1 \\ -1 & 0 \\ -1 & 1 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} > 0$$

```
X = list(np.array([[2,1], [-1, 0],
                   [-1, 1], [2, 2]]))
w = np.array([0, 0])
print ('w=', w)
change = True
while (change):
    change = False
    for x in X:
        if np.dot(x, w) <= 0:
            print ('w=', w, 'x=', x, '-')
            w += x
            change = True
                print ('w=', w)
```

```
w=[ 0 0]
w=[ 0 0] x= [ 2 1] -
w=[ 2 1] x= [-1 0] -
w=[ 1 1] x= [-1 1] -
w=[ 0 2] x= [-1 0] -
w=[-1 2] x= [ 2 1] -
w=[ 1 3] x= [-1 0] -
w=[ 0 3] x= [-1 0] -
w=[-1 3]
```

Реализация в scikit-learn

`sklearn.linear_model.Perceptron`

penalty=None	**Тип регуляризации**
alpha=0.0001	**Коэффициент регуляризации**
C=1.0	**Обратная величина к коэффициенту регуляризации**
fit_intercept=True	**Свободный член**
max_iter=1000	**Число итераций**
tol=0.001	**Критерий остановки**
shuffle=True	**Перемешивать ли данные**
verbose=0	**Вывод служебной информации**
eta0=1.0	**Темп сходимости**
early_stopping=False	**Можно отложит выборку и остановить настройку, когда нет уменьшения ошибки на ней**
validation_fraction=0.1	**Объём выборки для ES**
n_iter_no_change=5	**Сколько итераций ждать для ES**

`n_jobs=None, random_state=0, class_weight=None, warm_start=False`

Реализация в scikit-learn

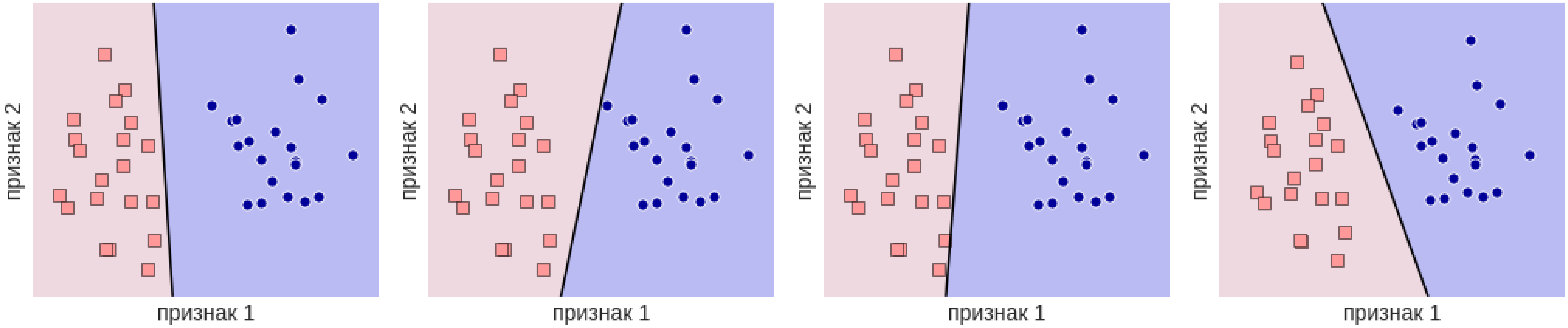
`sklearn.linear_model.SGDClassifier`
`sklearn.linear_model.SGDRegressor`

– общая реализация линейного алгоритма, обучающегося градиентным спуском работает с разреженными данными!

`loss="hinge"`	**Функция ошибки, варианты:**
	`"hinge"` – **SVM**
	`"log"` – **логистическая регрессия**
	`"modified_huber"`
	`"squared_hinge"`
	`"perceptron"`– **персептрон**
	для регрессии:
	`"squared_loss"`
	`"huber"`
	`"epsilon_insensitive"`
	`"squared_epsilon_insensitive"`

Support Vector Machine (SVM)

Метод опорных векторов – самый популярный метод 1990х

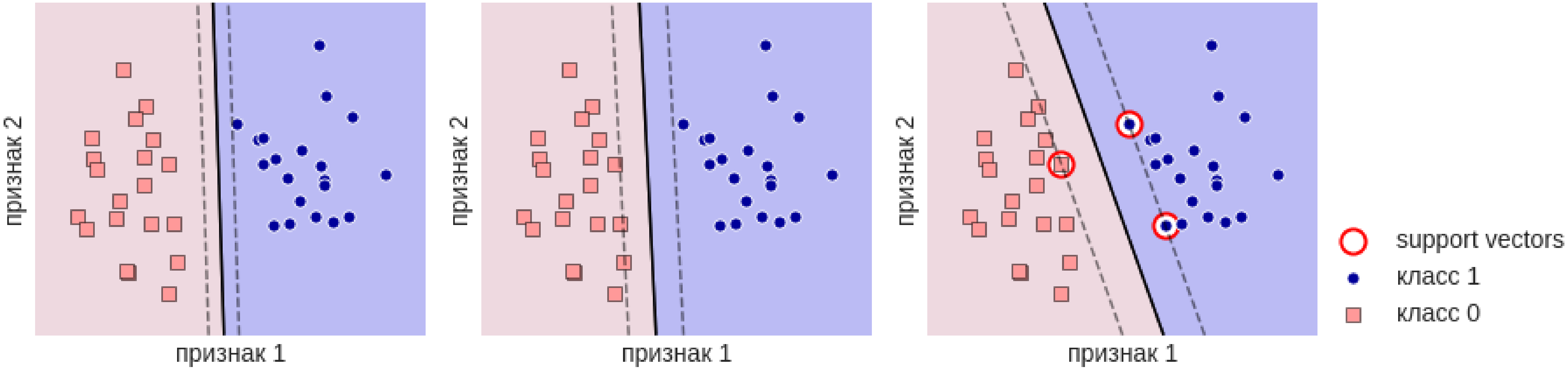


до сих пор пытались просто разделить точки…

а какой линейный классификатор лучше?

здесь для начала предполагаем, что классы линейно разделимы

SVM: идея максимального зазора



Идея:

если немного ошибёмся с коэффициентами / если объект задан неточно, всё равно решение должно быть правильным

Построение SVM эквивалентно нахождению кратчайшего отрезка, соединяющего выпуклые оболочки двух классов

SVM: постановка задачи

Пусть обучающая выборка:

$$\{(x_i, y_i)\}_{i=1}^m$$

$$y_i \in Y = \{\pm 1\}$$

Хотим разделить точки двух разных классов гиперплоскостью

$$a(x) = \mathrm{sgn}(w^{\mathrm{T}} x + b)$$

(здесь не вводим фиктивный константный признак)

Должно быть

$$w^{\mathrm{T}} x_i + b \geq +1 \text{ если } y_i = +1$$
$$w^{\mathrm{T}} x_i + b \leq -1 \text{ если } y_i = -1$$

(из-за нормировки это возможно)

Другая форма записи:

$$y_i (w^{\mathrm{T}} x_i + b) \geq 1$$

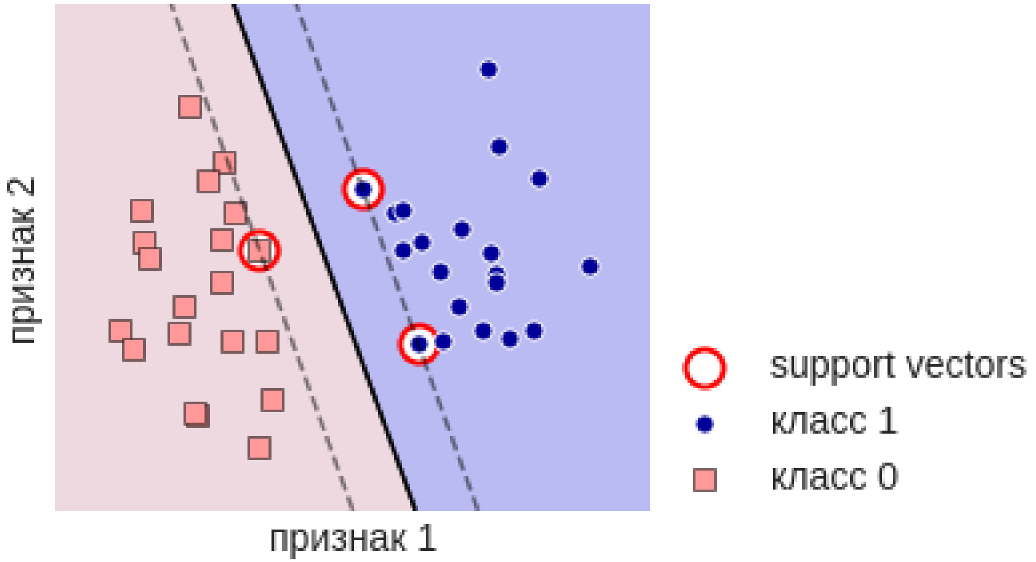
можно считать (из-за нормировки)**, что**

$$\min_i | w^{\mathrm{T}} x_i + b | = 1$$

SVM: постановка задачи

Расстояние от точки до гиперплоскости:

$$\rho(x_i, w^{\mathrm{T}}x + b) = \frac{|w^{\mathrm{T}}x_i + b|}{\|w\|}$$



хотим, чтобы минимум из этих расстояний был максимален

нормированный зазор (margin) –

$$\min_i \frac{|w^{\mathrm{T}}x_i + b|}{\|w\|} = \frac{1}{\|w\|} \to \max$$

Зазор (margin)

В общем случае, когда $X=\mathbb{R}^n, Y=\{\pm 1\}$

обучающая выборка: $\{(x_i, y_i)\}_{i=1}^m$

В общем случае
алгоритм со скоринговой функцией (score function):

$$a(x)=\operatorname{sgn}(b(x)),\ b(x)\in\mathbb{R}$$

$|b(x_i)|$ – **уверенность в ответе**

$y_i b(x_i)$ – **зазор** (насколько уверенность оправдала ожидание)

$\operatorname{sgn}(\cdot)$ – **деформация**

SVM: постановка задачи

задача квадратичного программирования (QP = Quadratic Program)
с *m* ограничениями (constraints)

$$\frac{\|w\|^2}{2} \to \min$$
$$y_i(w^\mathrm{T} x_i + b) \geq 1,\ i \in \{1, 2, \ldots, m\}$$

Заметим, что здесь также, как при регуляризации линейной регрессии, хотим квадрат нормы весов сделать меньше

«квадрат» – для удобства оптимизации

Заметим, что решение существует и единственно
Есть много солверов…

оффтопик: Оптимизация с ограничениями

$$f(w) \to \min$$
$$g_i(w) \le 0,\ i \in I,$$
$$h_j(w) = 0,\ j \in J.$$

Выпишем Лагранжиан

$$L(w,\alpha,\beta) = f(w) + \sum_{i \in I} \alpha_i g_i(w) + \sum_{j \in J} \beta_j h_j(w)$$

$$\alpha = (\alpha_i)_{i \in I} \ge 0,\ \beta = (\beta_j)_{j \in J}.$$

Заметим, что

$$\max_{\alpha,\beta} L(w,\alpha,\beta)$$

обращается в бесконечность, если нарушено хотя бы одно ограничение (по g_i или h_j),

в противном случае, совпадает с $f(w)$

оффтопик: Оптимизация с ограничениями

Поэтому можно решать такую задачу:

$$\min_{w} \max_{\alpha,\beta} L(w,\alpha,\beta)$$

из-за выпуклости всех функций min и max можно переставлять

Условия Кунна-Таккера (Karush-Kuhn-Tucker (KKT) Conditions):

в оптимальной точке

$$\alpha_i g_i(w) = 0$$

SVM: решение строгое

$$\frac{\|w\|^2}{2} \to \min$$

$$1 - y_i(w^{\mathrm{T}} x_i + b) \le 0,\ i \in \{1, 2, \ldots, m\}$$

Вспоминаем оптимизацию с ограничениями:

$$\min_{w,b} \max_{\alpha \ge 0} L(w, b, \alpha)$$

$$L(w, b, \alpha) = \frac{w^{\mathrm{T}} w}{2} + \sum_{i=1}^{m} \alpha_i (1 - y_i(w^{\mathrm{T}} x_i + b))$$

тут будет дифференцируемость, но есть ограничения

SVM: решение строгое

$$\min_{w,b}\max_{\alpha\ge 0}\left[\frac{w^{\mathrm{T}}w}{2}+\sum_{i=1}^{m}\alpha_i(1-y_i(w^{\mathrm{T}}x_i+b))\right]$$

возьмём производные, приравняем к нулю

$$\frac{\partial L}{\partial w}=0\Rightarrow w=\sum_{i=1}^{m}\alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b}=0\Rightarrow \sum_{i=1}^{m}\alpha_i y_i=0$$

Таким образом, оптимальный вектор весов –
взвешенная сумма признаковых описаний объектов из обучения

аналогично происходит и при настройке персептрона и логистической регрессии…

SVM: переход к двойственной задаче

Задача квадратичного программирования

$$\max_{\alpha\geq 0}\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}} x_j$$

при условиях $\sum_{i=1}^{m}\alpha_i y_i=0$

Информацию об описаниях объектов мы используем лишь в виде их попарных скалярных произведений!

когда решим задачу…

$$w=\sum_{i=1}^{m}\alpha_i y_i x_i$$

$$b=-\frac{1}{2}\left(\min_{i:\,y_i=+1} w^{\mathrm{T}}x_i+\max_{i:\,y_i=-1} w^{\mathrm{T}}x_i\right)$$

(связь между решением прямой и обратной задачи)

Условия Кунна-Таккера

рассмотрим

$$w=\sum_{i=1}^{m}\alpha_i y_i x_i$$

для полученного решения:

$$\alpha_i(1-y_i(w^{\mathrm{T}}x_i+b))=0$$

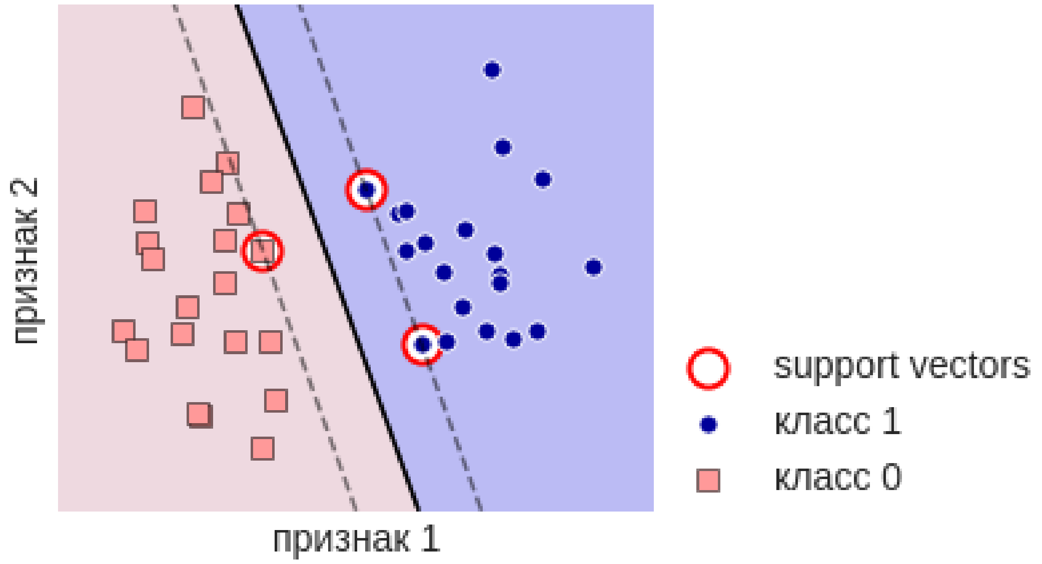
если $\alpha_i>0$**,**

то x_i **– опорный вектор (support vector)**

лежит на границе $y_i(w^{\mathrm{T}}x_i+b)=1$

большинство α_i **обратиться в ноль**

(из-за условий Кунна-Таккера)



Замечание о двойственной задаче

$$-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}} x_j \sim \alpha^{\mathrm{T}} \underbrace{\| y_i y_j x_i^{\mathrm{T}} x_j \|_{m\times m}}_{H} \alpha$$

матрица H **конгруэнтна (**$H = P^{\mathrm{T}}GP$**) матрице Грамма и является неотрицательно определённой**

Существует решение задачи оптимизации, но, вообще говоря, неединственное
(в отличие от прямой задачи)

Метод опорных векторов: SVM

Построить $H = \| y_i y_j x_i^{\mathrm{T}} x_j \|_{m\times m}$

Решить

$$-\frac{1}{2}\alpha^{\mathrm{T}} H\alpha + \tilde{1}^{\mathrm{T}}\alpha \to \max$$

при условиях

$$\alpha \geq 0,\ y^{\mathrm{T}}\alpha = 0$$

здесь везде векторная запись

Решение

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$S = \{i \in \{1,2,\ldots,m\} \mid \alpha_i > 0\}$$

$$b = \frac{1}{|S|}\sum_{i\in S}\left(y_i - w^{\mathrm{T}} x_i\right)$$

другой способ получения b

Зачем переходить к двойственной задаче

• **размерности**

м.б. удобно решать

выгодно, если признаковое пространство большое

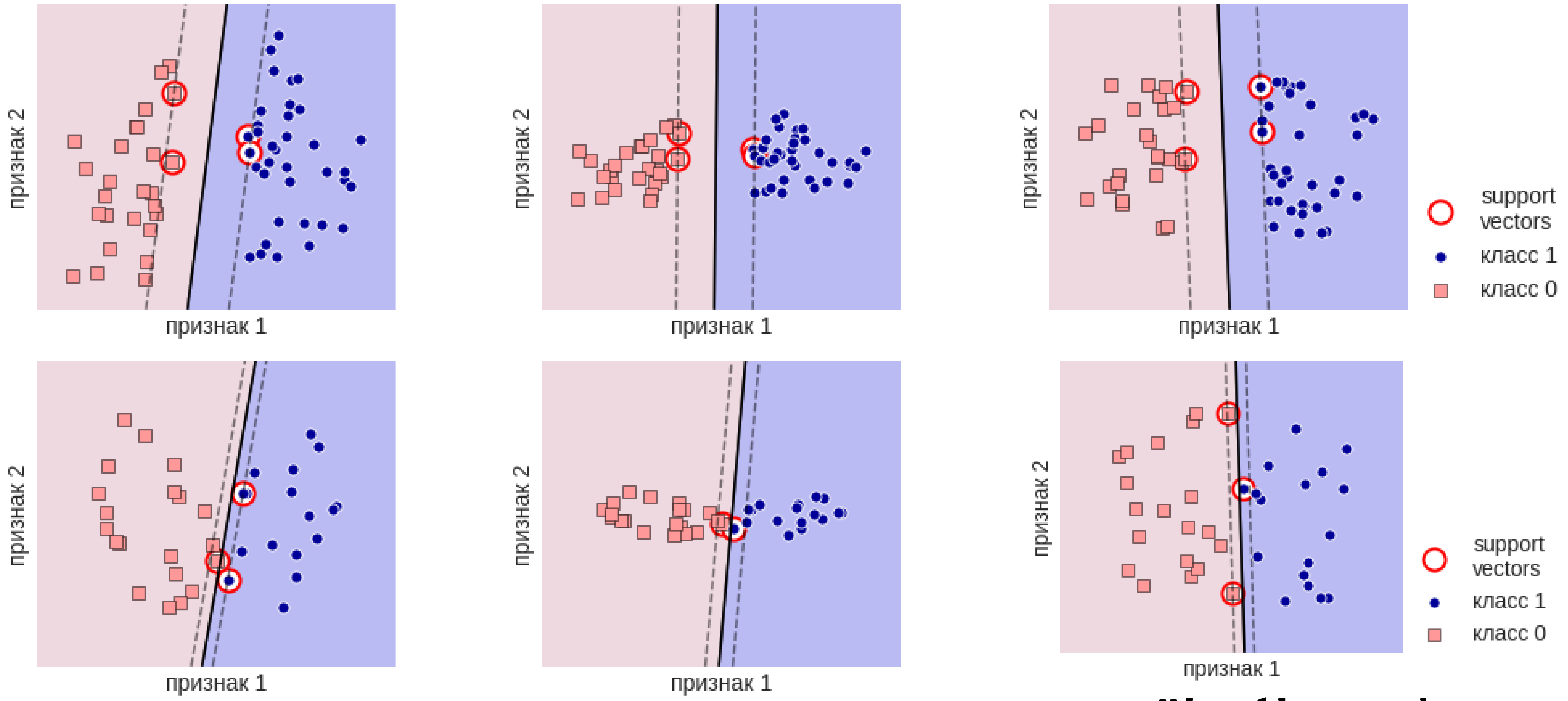
• **известная задача**

можно использовать солверы из готовых библиотек

• **возникли попарные произведения**

потом используем для kernel tricks

Чувствительность к масштабу и шумам



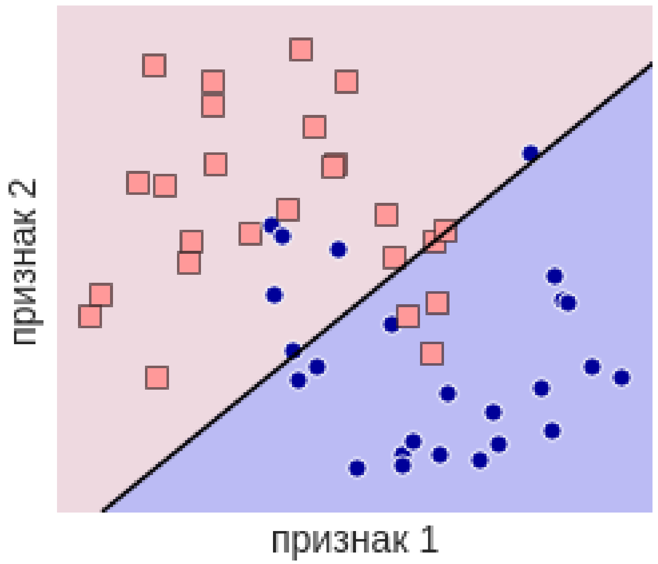
X[:, 1] / 10

X[:, 1] = rand

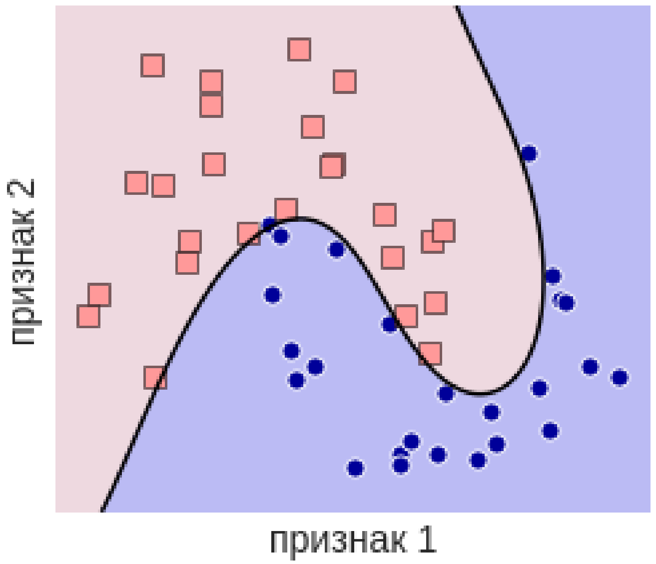
Если нет линейной разделимости

Два подхода (часто используются вместе)

1) разделять так, чтобы ошибок было мало



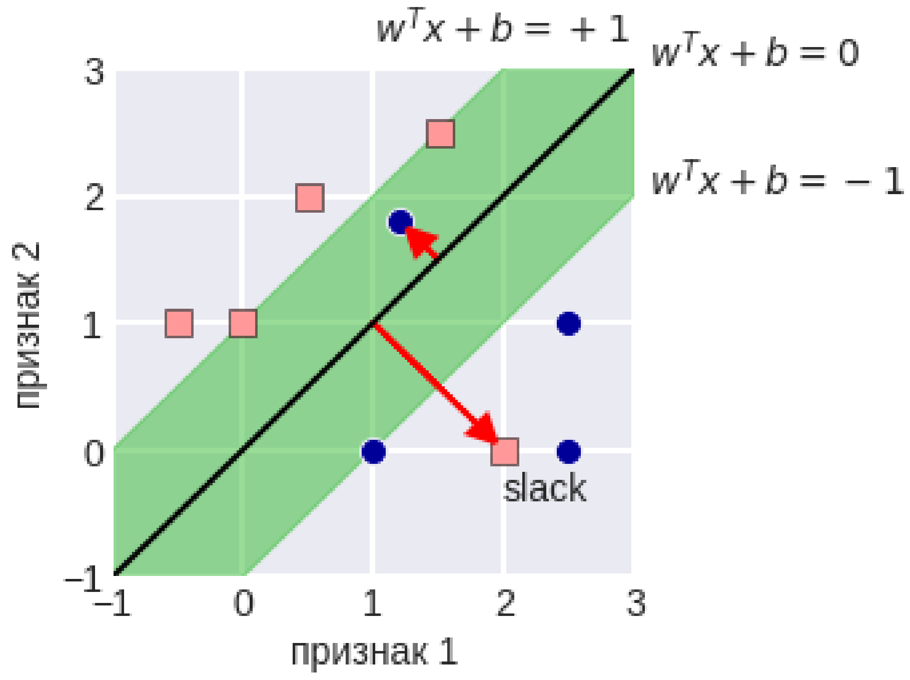
2) использование нелинейных разделяющих поверхностей



переход в другое признаковое пространство

потом подробно разберём!

Soft-Margin SVM: разделение допуская ошибки



позволить объектам «залезать» на половину другого класса

$$y_i(w^{\mathrm{T}} x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

но не хотим, чтобы было много больших «залезаний»

«slack variables»

Soft-Margin SVM: разделение допуская ошибки

Прямая задача:

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \ge 1 - \xi_i,\ \xi_i \ge 0,\ i \in \{1, 2, \ldots, m\}$$

Можно было бы рассматривать задачу с таким ограничением

$$\sum_{i=1}^{m}\xi_i \le C$$

тоже задача QP, но в два раза больше ограничений

C **– баланс между оптимизацией зазора и ошибки на обучении**

Если $C = 0$**, то получим решение** $w = \tilde{0}$**.**

Если $C \to +\infty$**, то получим решение как в «hard-margin objective»**

Soft-Margin SVM: двойственная задача

Прямая задача:

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0,$$
$$i \in \{1, 2, \ldots, m\}$$

Двойственная задача:

$$\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}} x_j \to \max_{0\leq\alpha\leq C}$$

появляется лишь ограничение $\alpha \leq C$

одно нетривиальное ограничение $\sum_{i=1}^{m}\alpha_i y_i = 0$

ДЗ вывести!

Получается, что Soft-Margin SVM решается аналогично…

Метод опорных векторов (SVM) в линейно неразделимом случае

Построить $H = \| y_i y_j x_i^{\mathrm{T}} x_j \|_{m\times m}$

Решить

$$-\frac{1}{2}\alpha^{\mathrm{T}} H\alpha + \tilde{1}^{\mathrm{T}}\alpha \to \max$$

при условиях

$$0 \le \alpha \le C,\ y^{\mathrm{T}}\alpha = 0$$

здесь везде векторная запись

Решение

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$S = \{i \in \{1,2,\ldots,m\} \mid 0 < \alpha_i \le C\}$$

$$b = \frac{1}{|S|}\sum_{i\in S}\left(y_i - w^{\mathrm{T}} x_i\right)$$

Soft-Margin SVM: численное решение

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i \in \{1, 2, \ldots, m\}$$

преобразуем:

$$\begin{cases} \xi_i \geq 1 - y_i(w^{\mathrm{T}}x_i + b) \\ \xi_i \geq 0 \end{cases} \Rightarrow \quad \xi_i \geq \max[1 - y_i(w^{\mathrm{T}}x_i + b),\ 0],$$

т.к. минимизируем сумму берём минимально возможное:

$$\xi_i = \max[1 - y_i(w^{\mathrm{T}}x_i + b),\ 0]$$

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

Soft-Margin SVM: численное решение

$$\underbrace{\frac{\|w\|^2}{2}}_{L_2-\text{регуляризация}} + C\underbrace{\sum_{i=1}^{m}\max[1-y_i(w^{\mathrm{T}}x_i+b),\,0]}_{\text{Hinge Loss}} \to \min$$

получили уже знакомое:

Hinge Loss + L_2-регуляризация

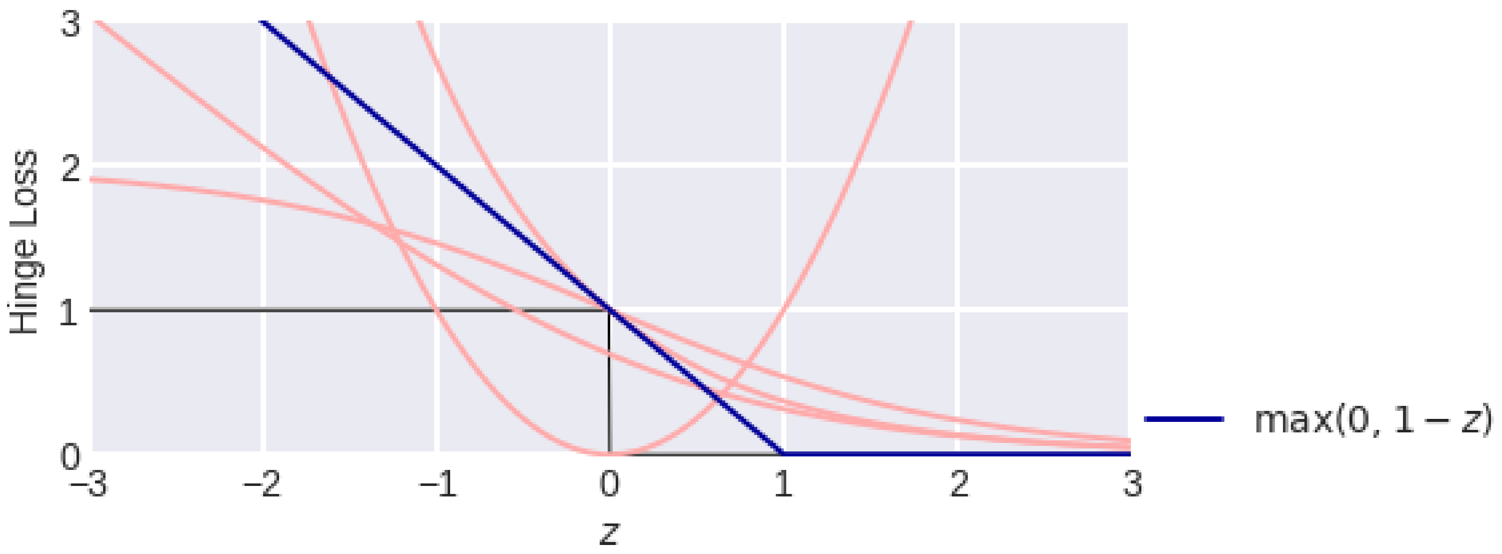
но тут нет дифференцируемости (в каждой точке) из-за max

Это точное решение поставленной задачи!

Понятно, почему в реализации SVM не коэффициент регуляризации, а (1 / коэф. регул.)

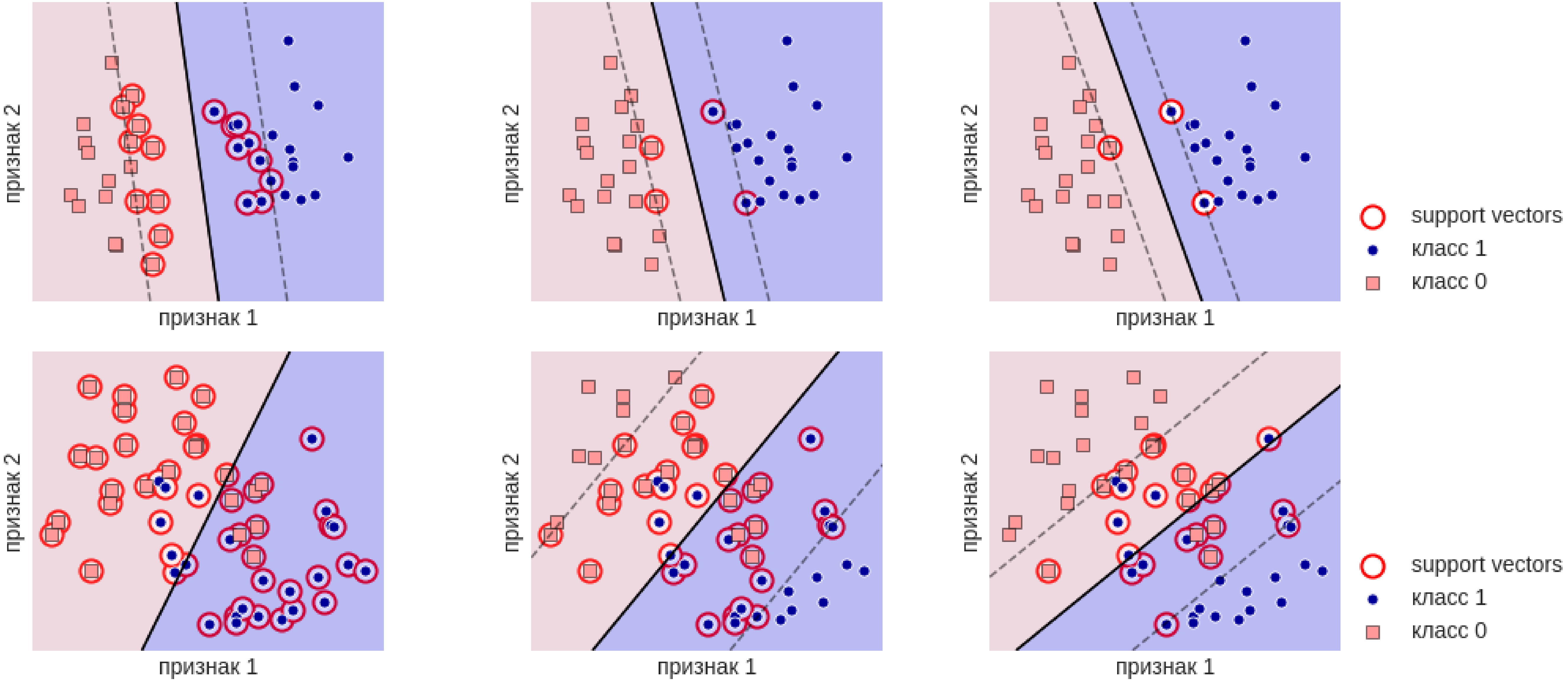
$$\mathrm{hinge}(y_i, w^{\mathrm{T}}x_i+b) + \frac{1}{2C}\|w\|^2 \to \min$$

Hinge loss



А в логистической регрессии:
логистическая функция ошибки + регуляризатор

Пример



$C = 0.01$ $C = 0.1$ $C = 1.0$

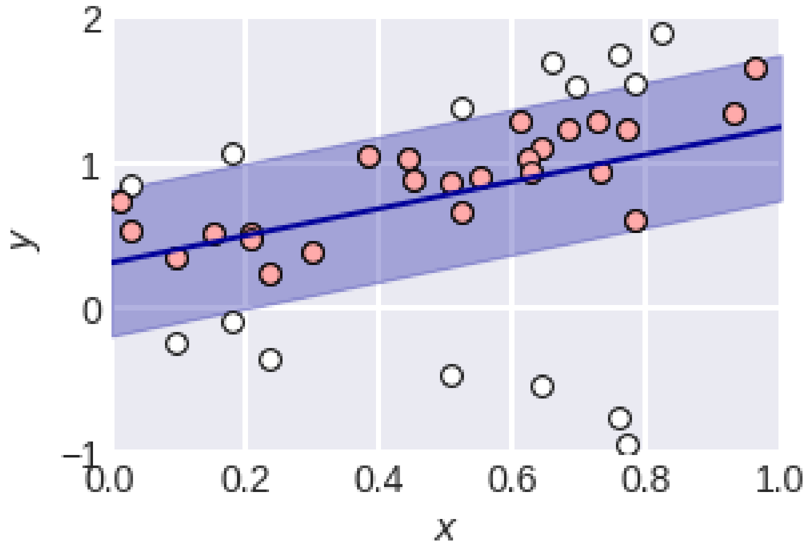
здесь опорными отмечены те объекты, которые пометила функция в sklearn

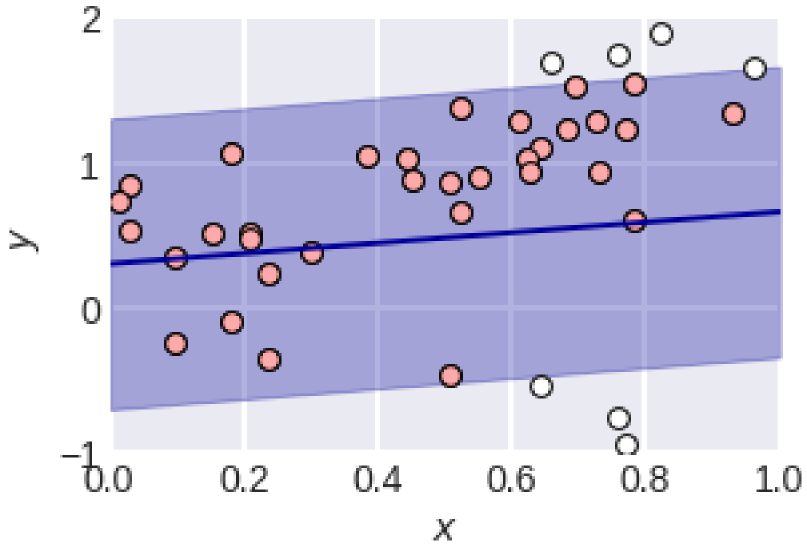
SVM Regression

хотим использовать регуляризацию и как можно лучше подстраиваться с ε-точностью:

$$\frac{\|w\|^2}{2} \to \min$$

$$|w^{\mathrm{T}} x_i + b - y_i| \le \varepsilon,\ i \in \{1, 2, \ldots, m\}$$





выполнение неравенства – попадание точки в полосу

сейчас формализуем – что мы хотим «от попадания»

SVM Regression

$$\frac{\|w\|^2}{2} \to \min$$

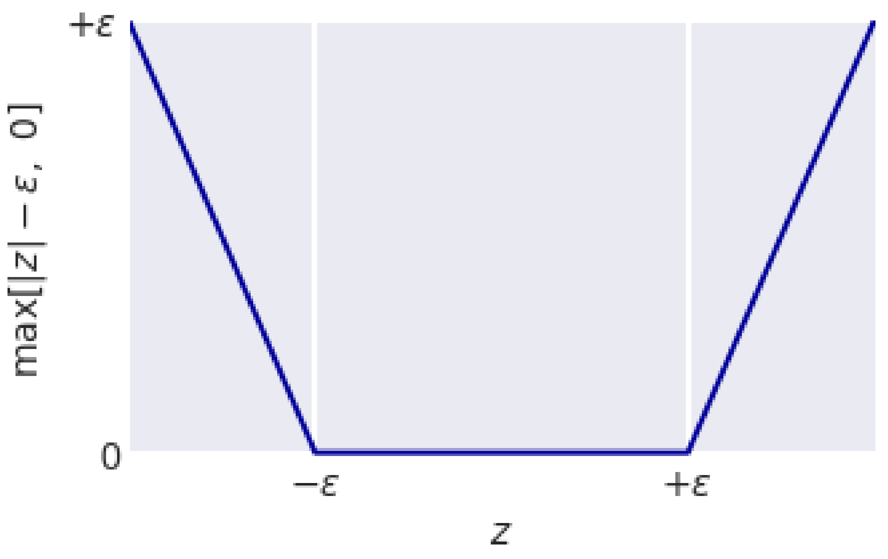
$$|w^{\mathrm{T}}x_i + b - y_i| \leq \varepsilon,\ i \in \{1, 2, \ldots, m\}$$

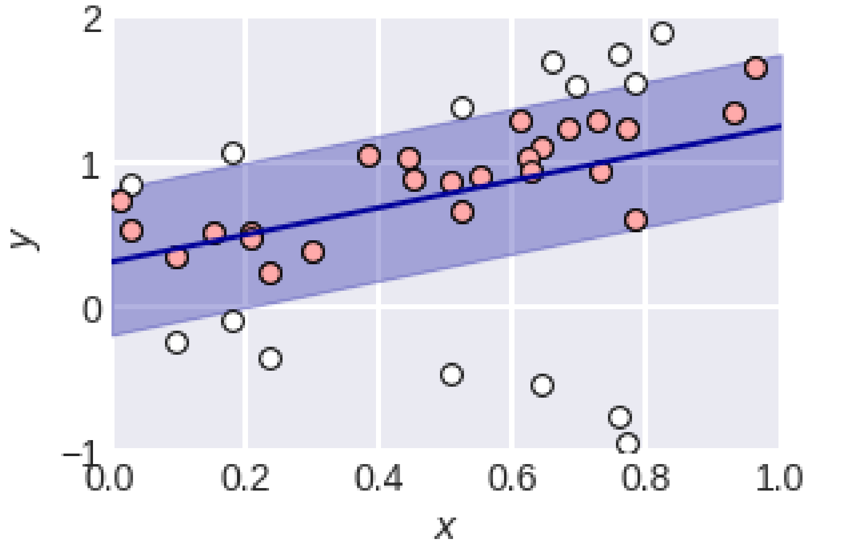
Записываем такую функцию ошибки:

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m} \max[|w^{\mathrm{T}}x_i + b - y_i| - \varepsilon, 0] \to \min$$

Решение будет зависеть от объектов, для которых ошибка превышает порог…

После замены переменных задача сводится к QP



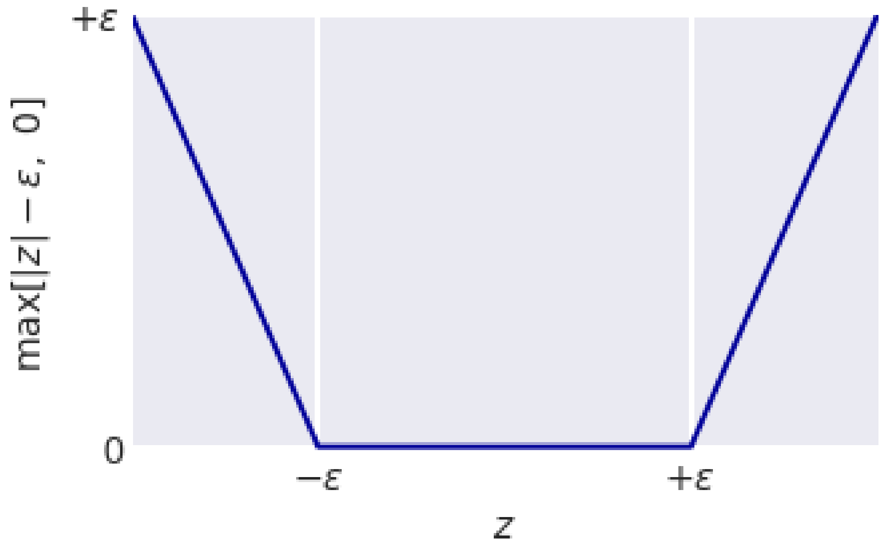


SVM Regression

$$\frac{\| w \|^2}{2} + C\sum_{i=1}^{m} \max[| w^{\mathrm{T}} x_i + b - y_i | - \varepsilon, 0] \to \min$$

ε-чувствительный Loss + регуляризатор

$$\sum_{i=1}^{m} \max[| w^{\mathrm{T}} x_i + b - y_i | - \varepsilon, 0] + \frac{1}{2C} \| w \|^2 \to \min$$



SVM Regression – переход к двойственной задаче

вводим штрафующие слагаемые

$$\xi_i^+ \geq 0,\ \xi_i^- \geq 0,\ i \in \{1,2,\ldots,m\}$$

для этого неравенства:

$$w^{\mathrm{T}} x_i + b - y_i \leq +\varepsilon + \xi_i^+$$
$$w^{\mathrm{T}} x_i + b - y_i \geq -\varepsilon - \xi_i^-$$

тогда наш функционал

$$\frac{\| w \|^2}{2} + C\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-) \to \min$$

дальше аналогично переходим к Лагранжиану… пропускаем это

SVM Regression – переход к двойственной задаче

$$\sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)(w^{\mathrm{T}}x_i + b - y_i) - \varepsilon\sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)x_i^{\mathrm{T}}x_j \to \max$$

$$0 < \alpha_i^+ \le C, 0 < \alpha_j^+ \le C, \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-) = 0$$

После решения QP

$$w = \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)x_i$$

$$S = \{i | 0 < \alpha_i^+ \le C, 0 < \alpha_i^- \le C, (\xi_i^+ = 0) \vee (\xi_i^- = 0)\}$$

$$b = \frac{1}{|S|}\sum_{i\in S}\left(y_i - w^{\mathrm{T}}x_i - \varepsilon\right)$$

Реализация в scikit-learn

`sklearn.svm.LinearSVC`

`penalty="l2"`	**Тип регуляризации**
`loss="squared_hinge"`	**Ошибка регуляризации, более традиционная** `"hinge"`
`dual=True`	**Какую задачу решать** `dual=False` **когда** `n_samples > n_features`
`C=1.0`	**Обратная величина к коэффициенту регуляризации**
`fit_intercept=True`	**Свободный член**
`intercept_scaling=1`	**Значение фиктивного признака**

```
tol=0.0001, multi_class="ovr", class_weight=None, verbose=0,
random_state=None, max_iter=1000
```

SVM

• **естественное определение «оптимальной разделяющей гиперплоскости»**

+ геометрическая интерпретация

+ некоторая защита от проклятия размерности

• **есть теоретическое обоснование (не всё рассказали)**

большой зазор $\Rightarrow$ меньше переобучение

• **при оптимизации нет проблем локальных минимумов**

• **решение определяется «опорными объектами»**

их сложнее всего классифицировать

– только их можно оставить в выборке

• **есть нелинейные модификации «kernel tricks»**

это, вообще говоря, не линейный метод!

будет дальше

SVM

• **должно быть хорошее пространство**
(однородные признаки в одной шкале)

• **тогда работают линейные SVM**
(нелинейные – с ядрами – успешно заменяются другими алгоритмами)

• **не подходят для больших данных**
(особенно нелинейные)

• **при нелинейности интерпретация немного теряется…**
иногда непонятно, в каком именно пространстве решается задача

• **опорные объекты – нельзя считать «базовыми»**
из-за этого было много споров,
работает ли метод «правильно»

Итоги

Линейный классификатор ~ разделение точек гиперплоскостью
Есть простые методы настройки

Нигде в линейном классификаторе не минимизировали число ошибок
NP-полная задача
Но заменяли эту функцию ошибок другой… суррогатной

SVM зависит от масштаба признаков!

SVM чувствителен к шуму (шумовым признакам и объектам)

SVM называют лучшим методом линейной классификации… спорно

Ссылки

Alexey Nefedov «Support Vector Machines: A Simple Tutorial»

https://svmtutorial.online/

Tristan Fletcher «Support Vector Machines Explained»

https://static1.squarespace.com/static/58851af9ebbd1a30e98fb283/t/58902fbae4fcb5398aeb7505/1485844411772/SVM+Explained.pdf