«Машинное обучение»

Линейные модели классификации

Александр Дьяконов

План

Линейные решающие модели в задаче бинарной классификации

Идея максимального зазора

Суррогатные функции ошибки

Метод опорных векторов (SVM)

Soft-Margin SVM: разделение допуская ошибки

SVM Regression

Логистическая регрессия

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)

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

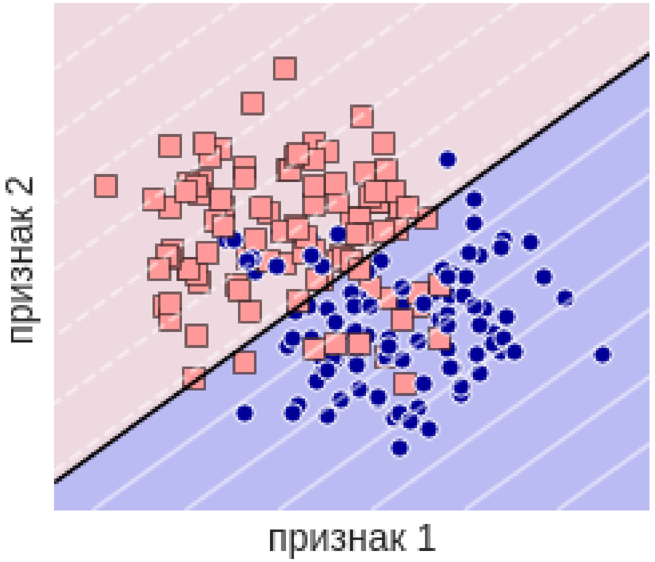
случай $w^{\mathrm{T}}x+b=0$ **нам тут не особо важен**

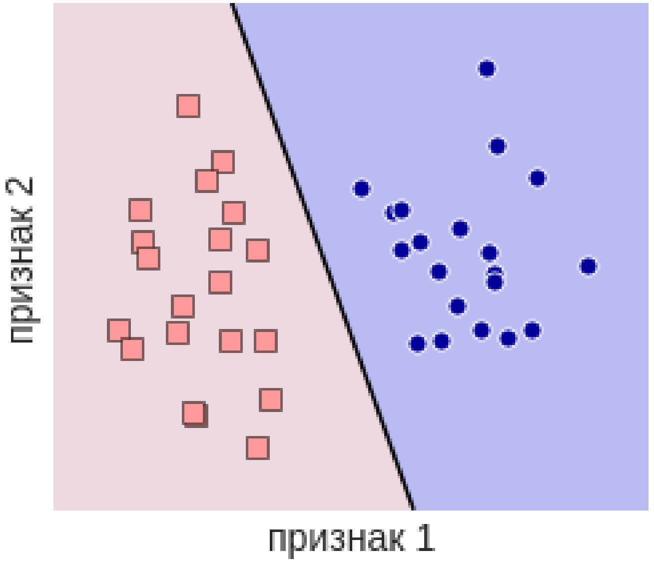
Линейный классификатор

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

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

$$L(X_{\text{train}}, a) = \sum_{t=1}^{m} L(y_t, a(x_t)) \to \min \qquad L(y_t, a(x_t)) = \begin{cases} 1, & \operatorname{sgn} w^{\text{т}} x_t \neq y_t \\ 0, & \operatorname{sgn} w^{\text{т}} x_t = y_t, \end{cases}$$

естественно минимизировать число ошибок, но

- **производная = 0 (не везде дифференцируема)**
- **выдаёт мало информации**
  только число ошибок, а не их «фатальность»
- **оптимизация здесь – NP-полная задача**

Зазор (Margin)

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

можно переписать:

$$L(y_t, a(x_t)) = I[y_t w^{\mathrm{T}} x_t \leq 0] = \theta(-z_t)$$

где $\theta(z) = I[z \geq 0]$

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

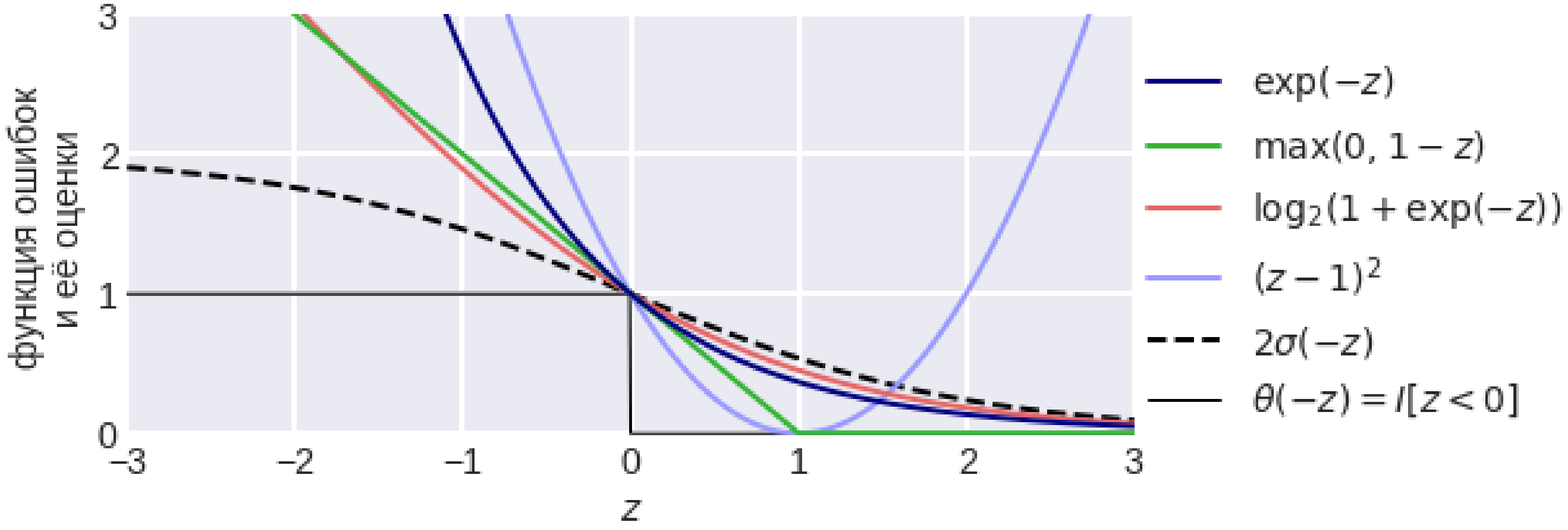
пропорционален расстоянию до ГП с точностью до $\|w\|$ (дальше), знак ~ ошибка

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

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**

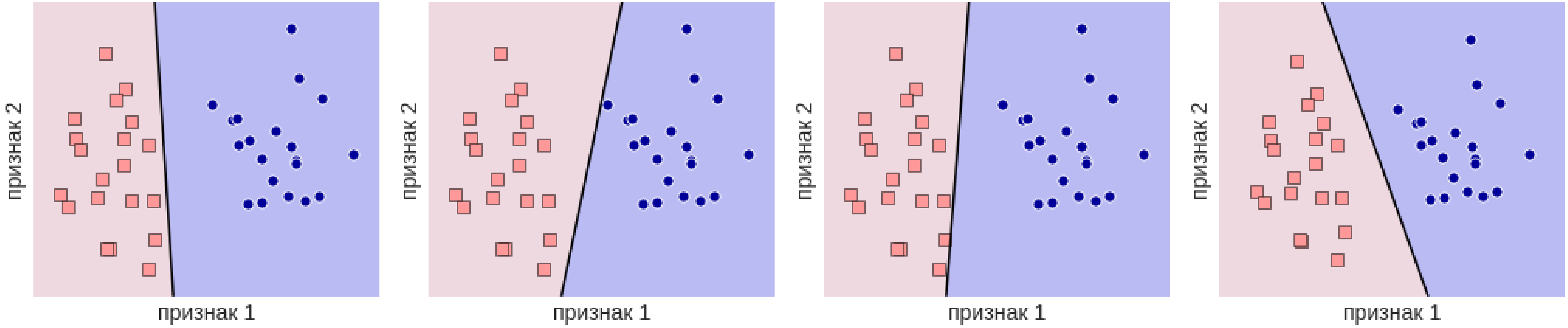
для регрессии:

`"squared_loss"`
`"huber"`
`"epsilon_insensitive"`
`"squared_epsilon_insensitive"`

Support Vector Machine (SVM)

Метод опорных векторов – самый популярный метод 1990х

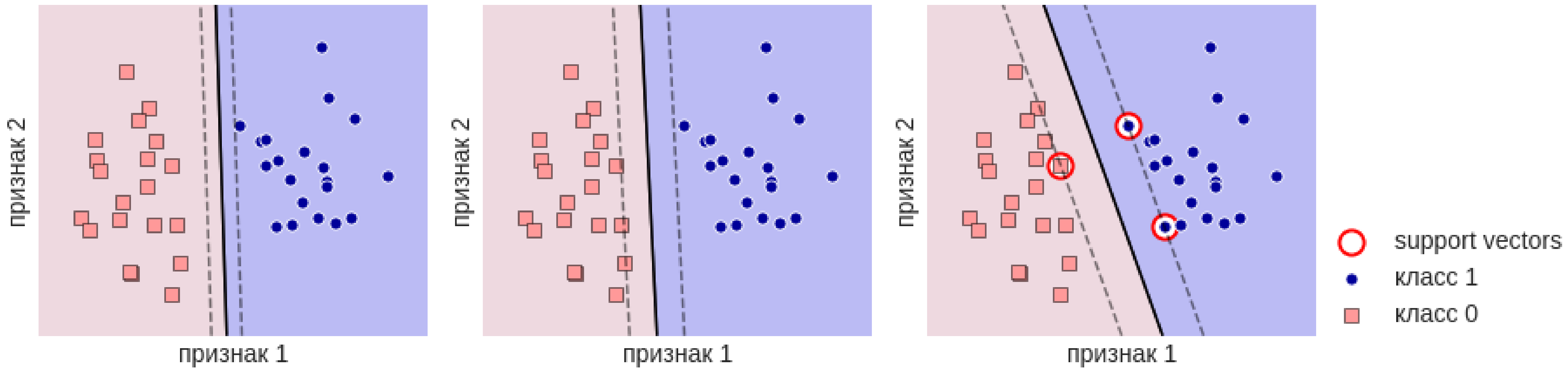
здесь для начала предполагаем, что классы линейно разделимы



до сих пор пытались просто разделить точки…

а какой линейный классификатор лучше?

SVM: идея максимального зазора



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)

коэффициенты нужны с точностью до множителя $\alpha > 0$**:**

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

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)

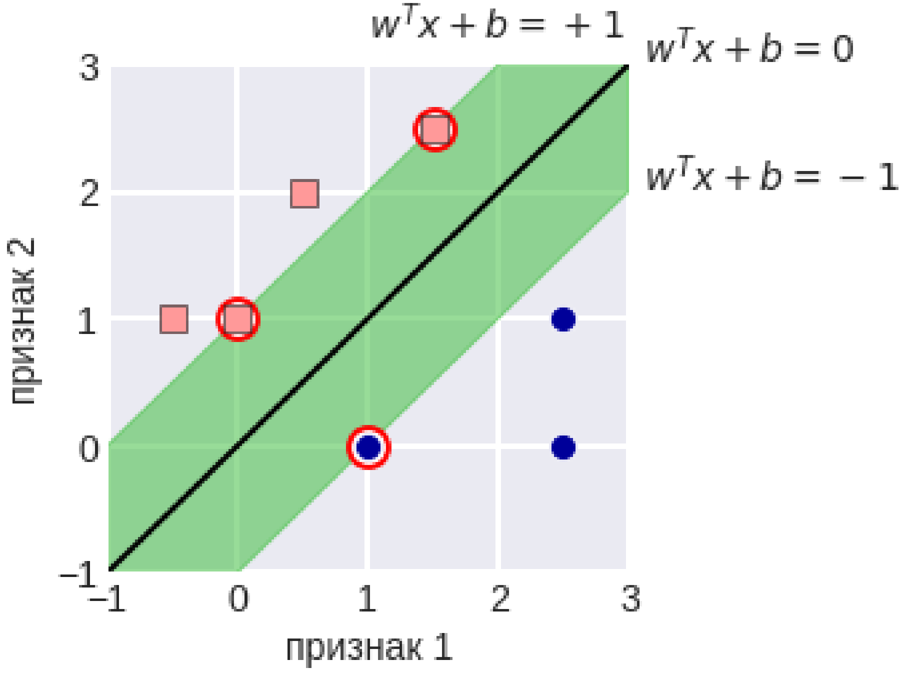
Другая форма записи:

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

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

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

SVM: постановка задачи



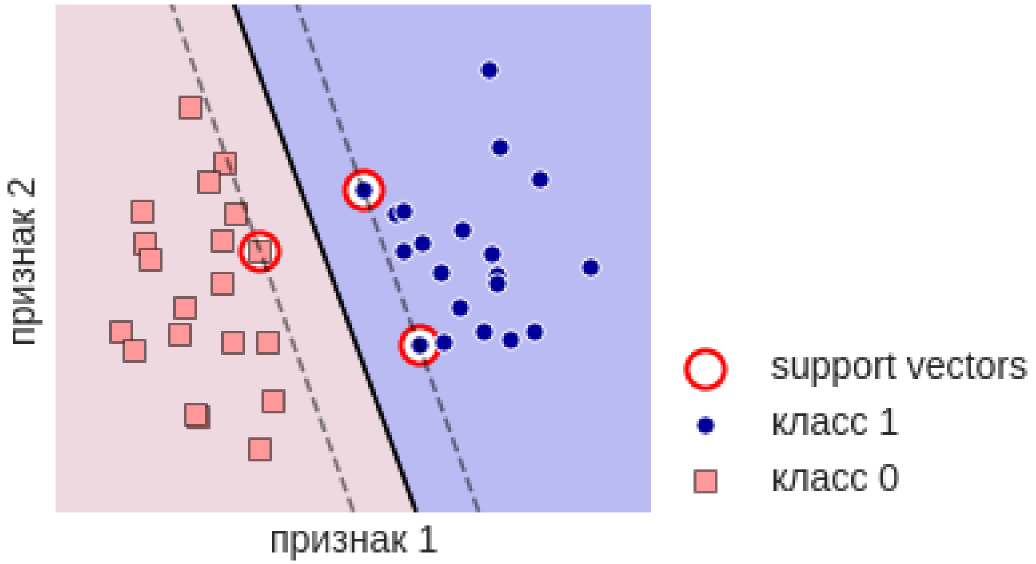
$$y_i(w^{\mathrm{T}} x_i + b) \geq 1$$

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

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)

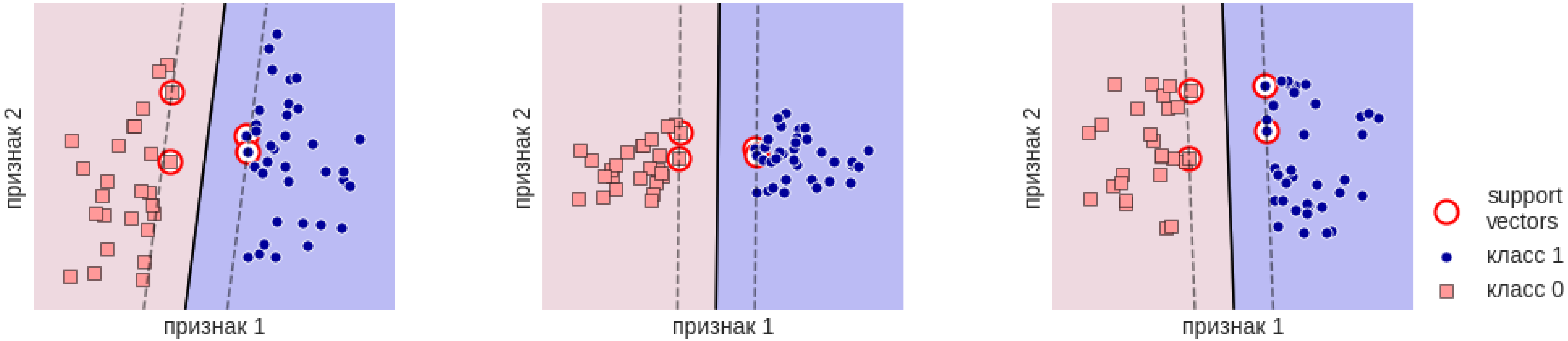
$$\frac{\| w \|^2}{2} \to \min$$
$$y_i (w^\mathrm{T} x_i + b) \geq 1,\ i \in \{1, 2, \ldots, m\}$$

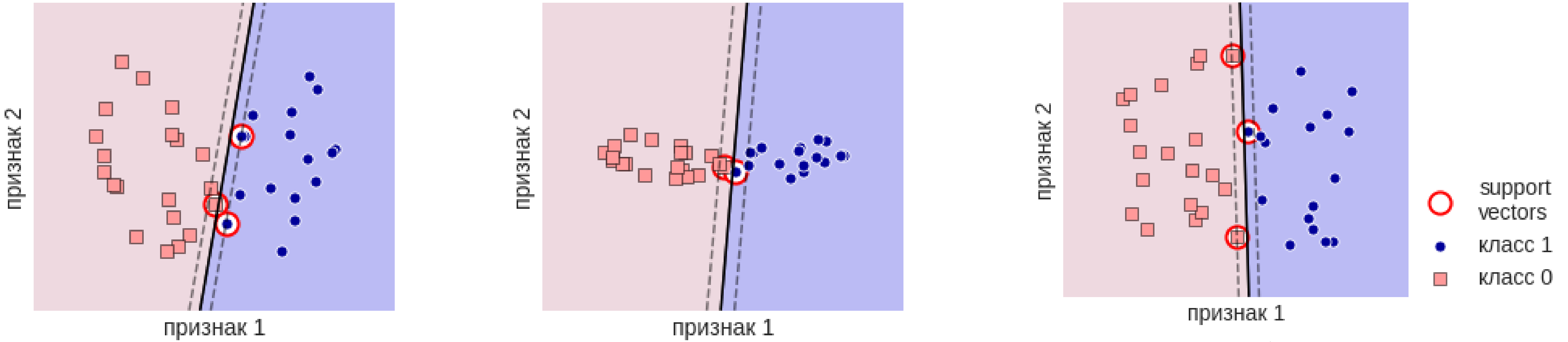
Заметим, что здесь также, как при регуляризации линейной регрессии, хотим квадрат нормы весов сделать меньше

«квадрат» – для удобства оптимизации

Заметим, что решение существует и единственно
Есть много солверов…

Чувствительность к масштабу и шумам





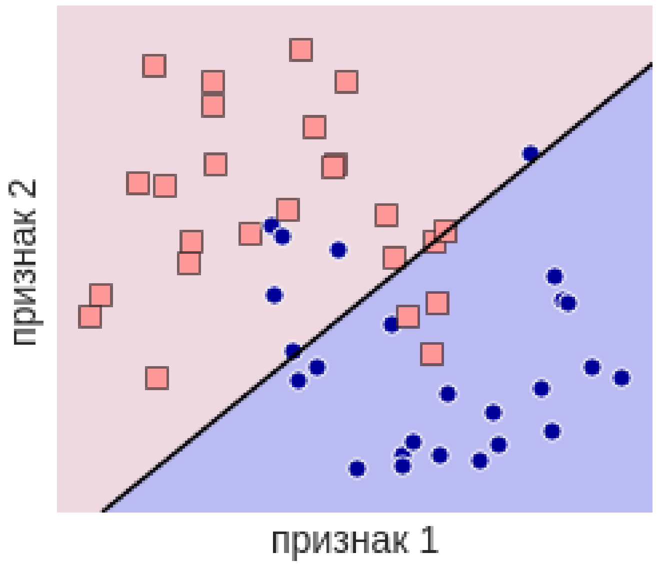
`X[:, 1] / 10`

`X[:, 1] = rand`

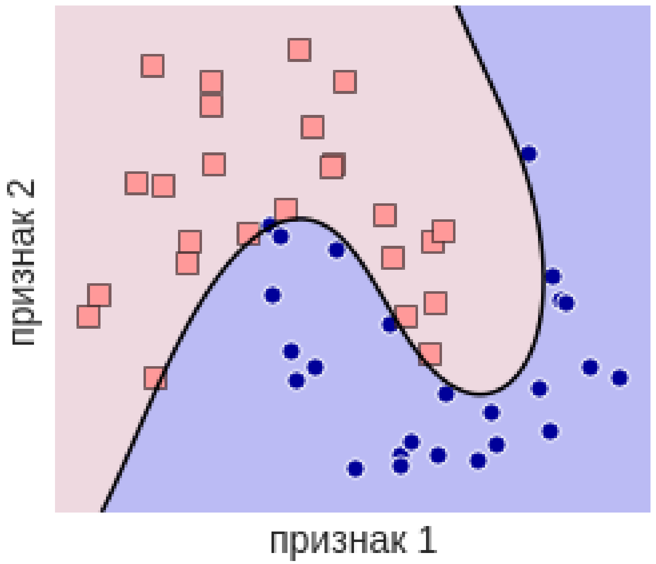
Если нет линейной разделимости

Два подхода (часто используются вместе)

1) разделять так, чтобы ошибок было мало

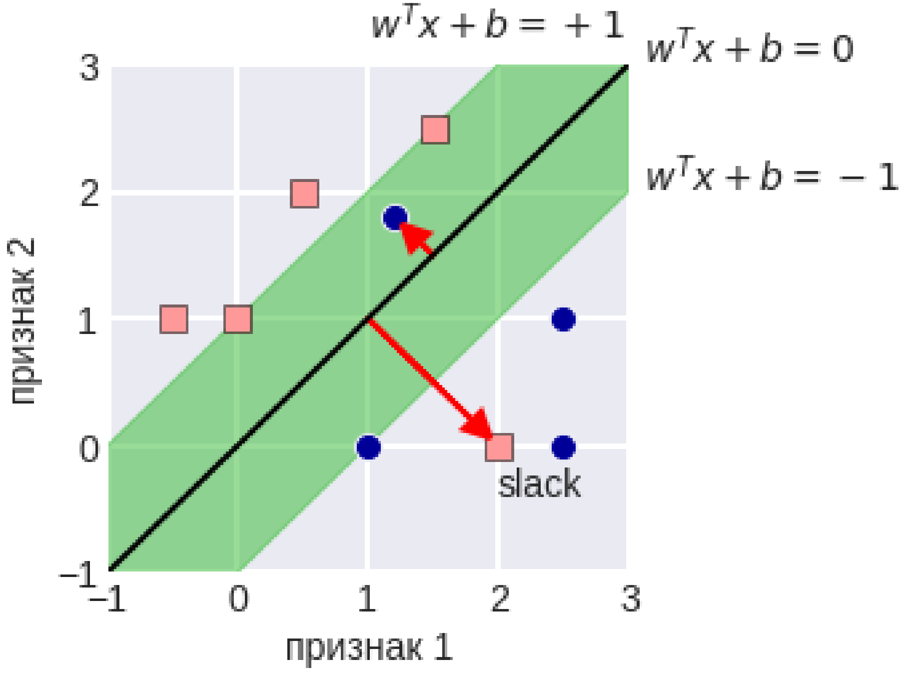


2) использование нелинейных разделяющих поверхностей



переход в другое признаковое пространство

Soft-Margin SVM: разделение допуская ошибки



позволить объектам «залезать» в полупространство 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»:

штрафующие переменные (slack variables)

Soft-Margin SVM: разделение допуская ошибки

Прямая задача:

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

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

тоже задача QP, но в два раза больше ограничений

Можно было бы рассматривать задачу с таким ограничением

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

понятно, что можно было бы и

$$+C\sum_{i=1}^{m}| \xi_i |^d$$

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: 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:

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

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

Поставленная задача свелась к …

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

получили уже знакомое: Hinge Loss + L_2-регуляризация

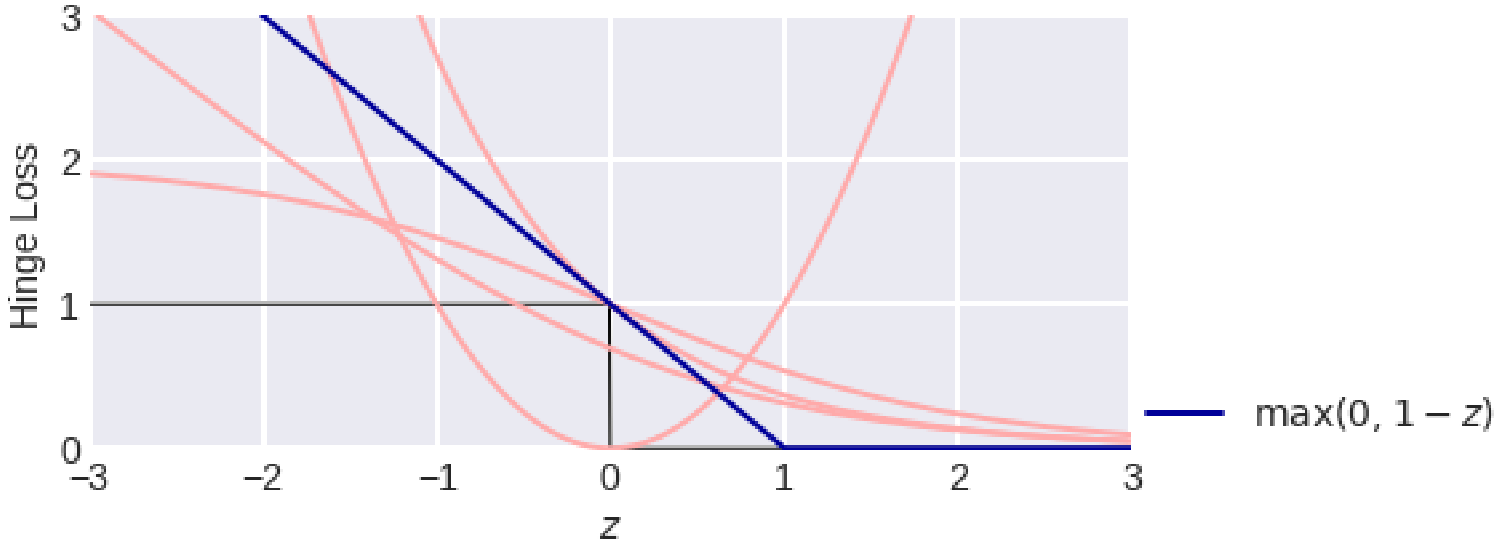
но тут нет дифференцируемости (в каждой точке) из-за max

Также видно, что если $1-y_i(w^{\mathrm{T}}x_i+b)\le 0$,

т.е. зазор $\ge$ 1 (достаточно большой), то объект не влияет на решение

решение зависит только от опорных объектов

Hinge loss

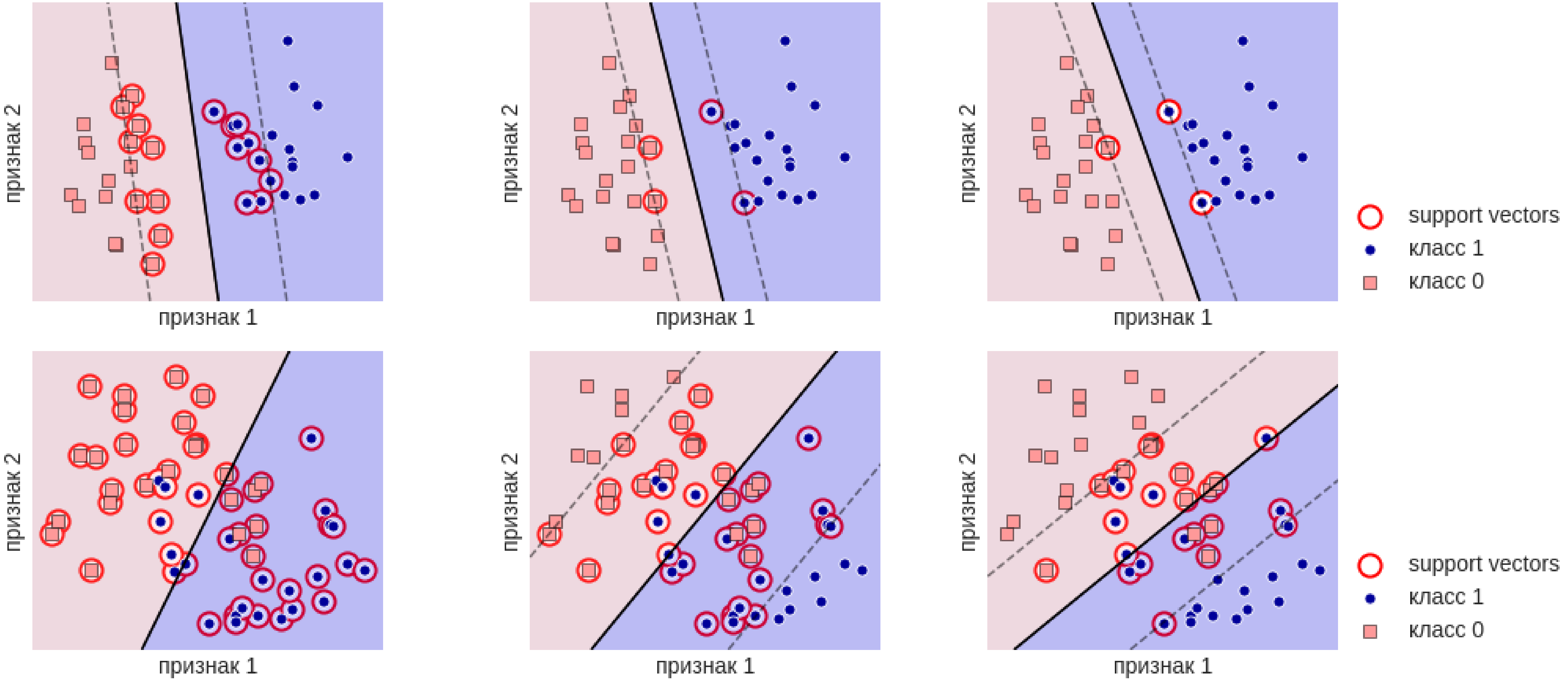


А в логистической регрессии: логистическая функция ошибки + регуляризатор

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

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

Пример



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

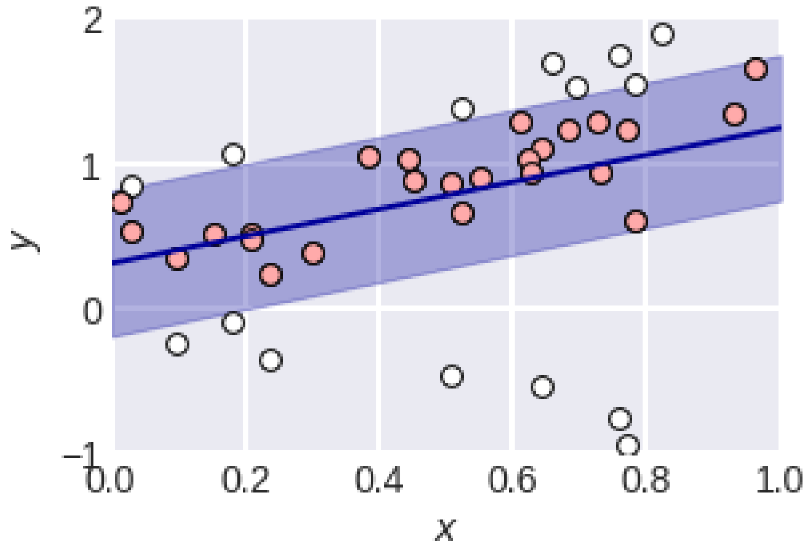
здесь опорными отмечены те объекты, которые пометила функция в sklearn

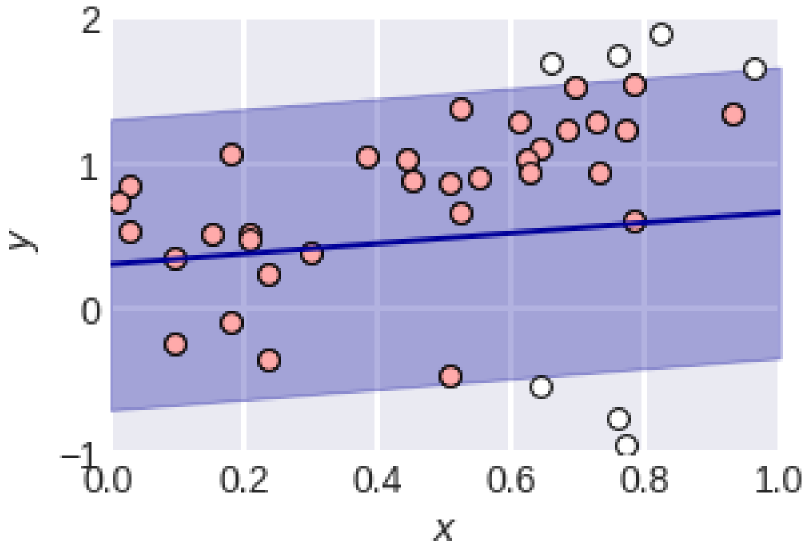
SVM Regression

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

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

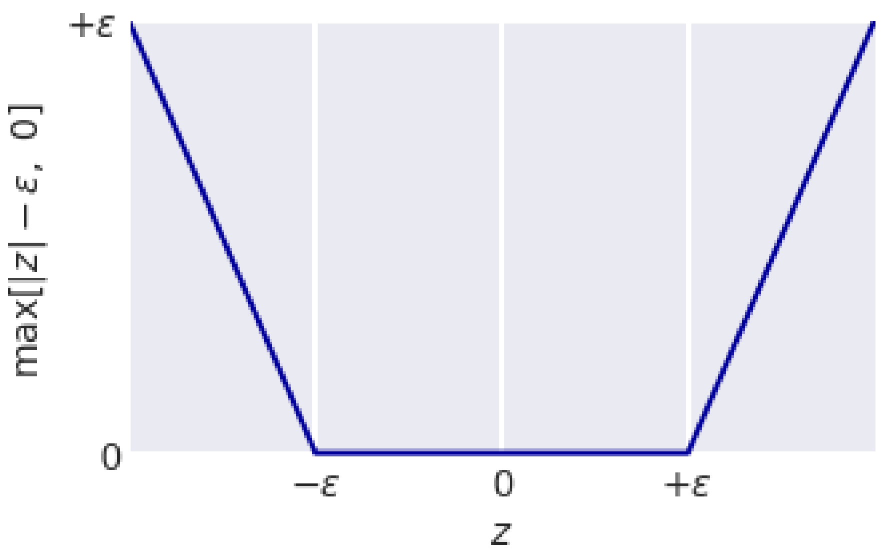
$$|w^{\mathrm{T}} x_i + b - y_i| \le \varepsilon,\ i \in \{1,2,\ldots,m\}$$

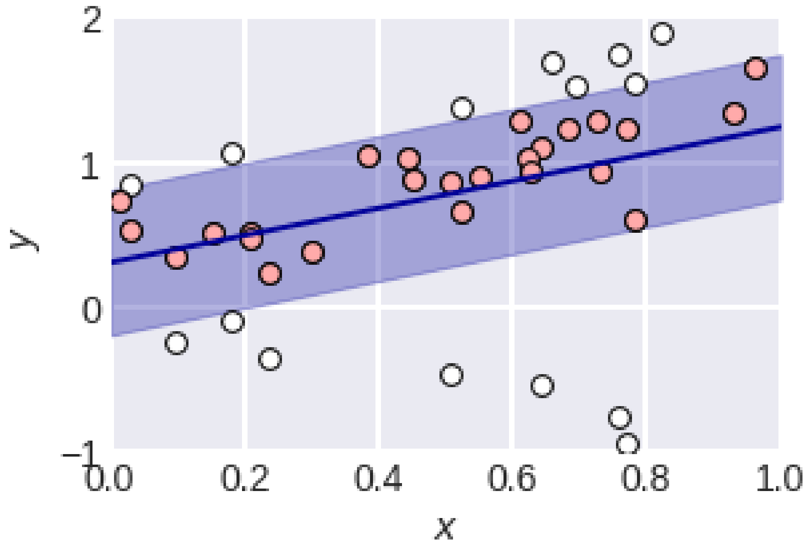
Записываем такую функцию ошибки:

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

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

После замены переменных задача сводится к QP



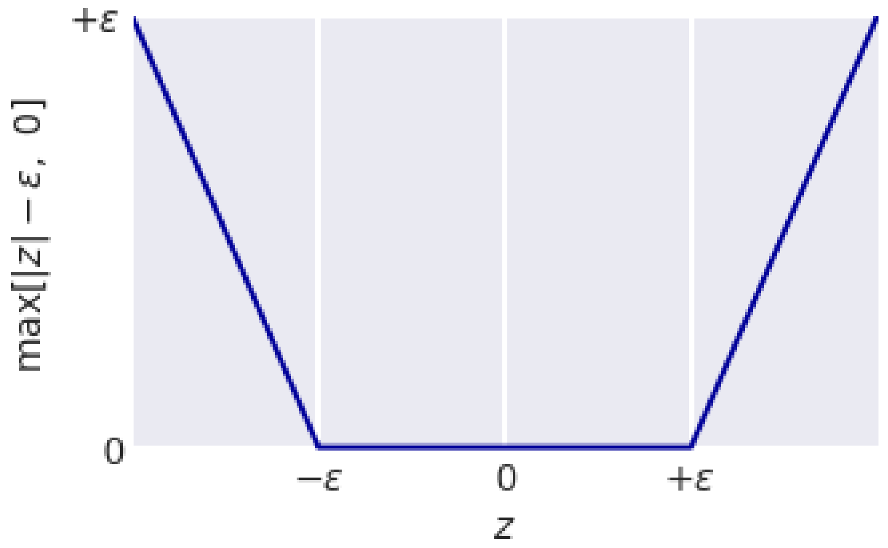


SVM Regression

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

ε-чувствительный Loss + регуляризатор

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



Реализация в scikit-learn

```
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

clf = make_pipeline(StandardScaler(),
                    LinearSVC(penalty='l2', # Тип регуляризации
                    loss='squared_hinge',   # Ошибка регуляризации, более
                                            # традиционная "hinge"
                    dual='warn', # Какую задачу решать,
                                 # dual=False когда n_samples > n_features
                    C=1.0, # Обратная величина к коэффициенту регуляризации
                    multi_class='ovr',      #  = one-vs-rest
                    fit_intercept=True,     # Свободный член
                    intercept_scaling=1)    # Значение фиктивного признака
clf.fit(X, y)
```

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!

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»

Линейные скоринговые модели в задаче бинарной классификации

Пусть $X=\mathbb{R}^n, Y=\{0,1\}$

Как решать задачи классификации с помощью линейной модели:
будем получать вероятность принадлежности к классу 1

$$a(x)\in[0,1]$$

Любая линейная функция на $\mathbb{R}^n$ будет получать значения в $\mathbb{R}$,
поэтому нужна деформация (transfer function):

$$\sigma:\mathbb{R}\to[0,1]$$

Решаем задачу так:

$$\sigma(w^{\mathrm{T}}x)$$

проекция на одномерное пространство и деформация

Функции деформации

В логистической регрессии

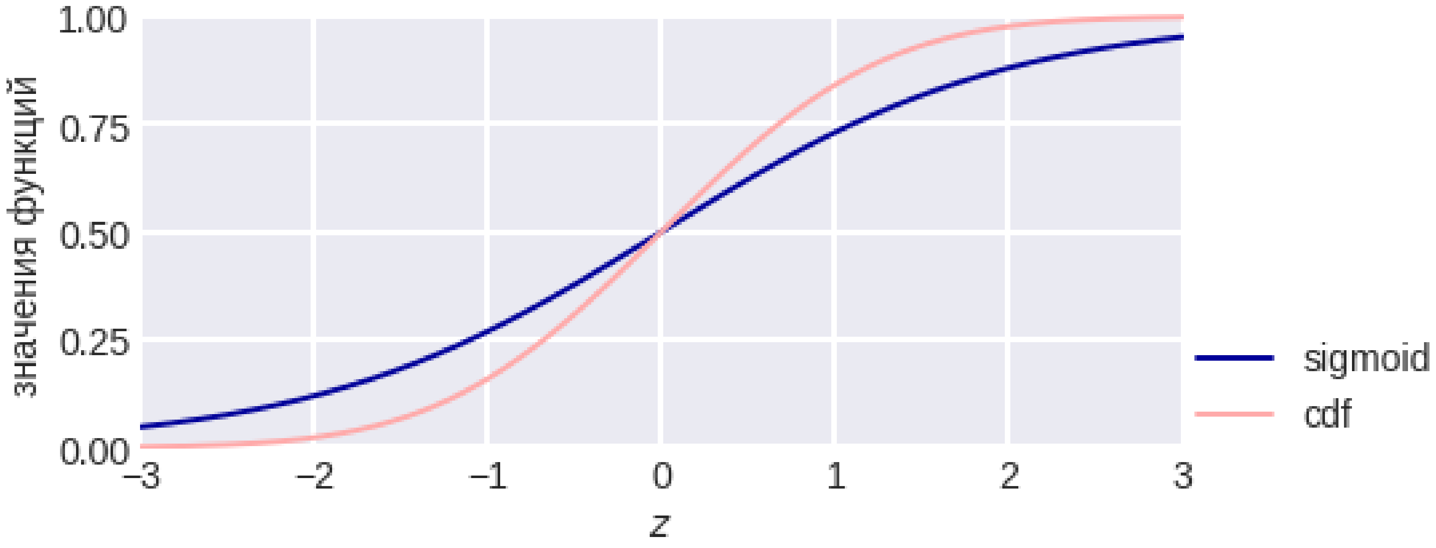
$$\sigma(z)=\frac{1}{1+e^{-z}}$$

Логистическая функция (сигмоида)

В Probit-регрессии

$$\Phi(z)=\frac{1}{\sqrt{2\pi}}\int\limits_{-\infty}^{z}\exp(-t^2/2)\partial t$$

Normal Cumulative distribution function



Логистическая регрессия

$$P(Y=1\,|\,x)=\sigma(z)=\frac{1}{1+e^{-z}}\in(0,1),$$

$$z=w^{\mathrm{T}}x=w_0+w_1X_1+\ldots+w_nX_n,$$

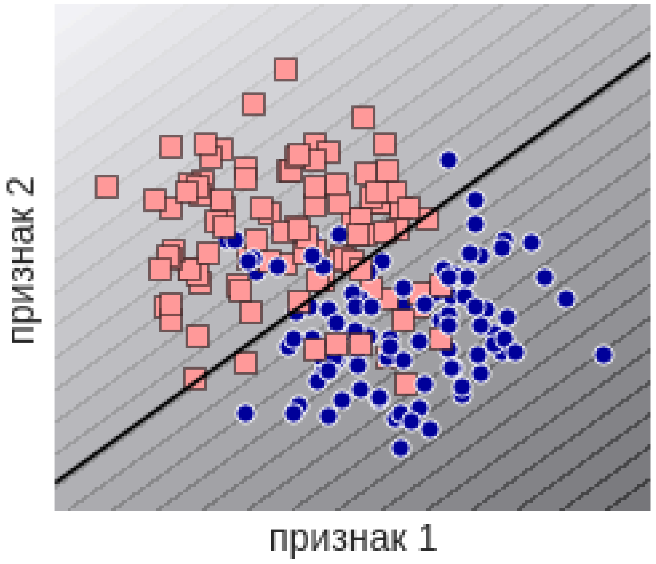
$$\log\left(\frac{\sigma(z)}{1-\sigma(z)}\right)=z$$

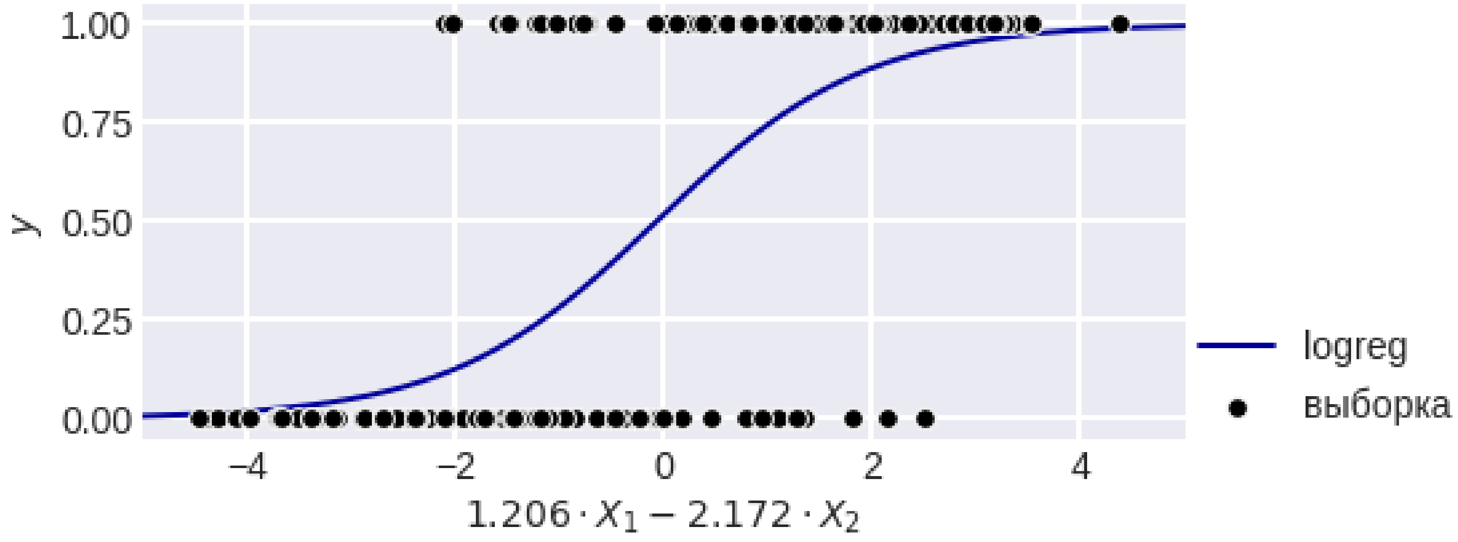
– монотонное преобразование, которое называют logit-transformation

Решаем задачу классификации, но метод называется логистическая регрессия

Геометрический смысл логистической регрессии

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X, y)
a = model.predict_proba(X_test)[:,1]
```

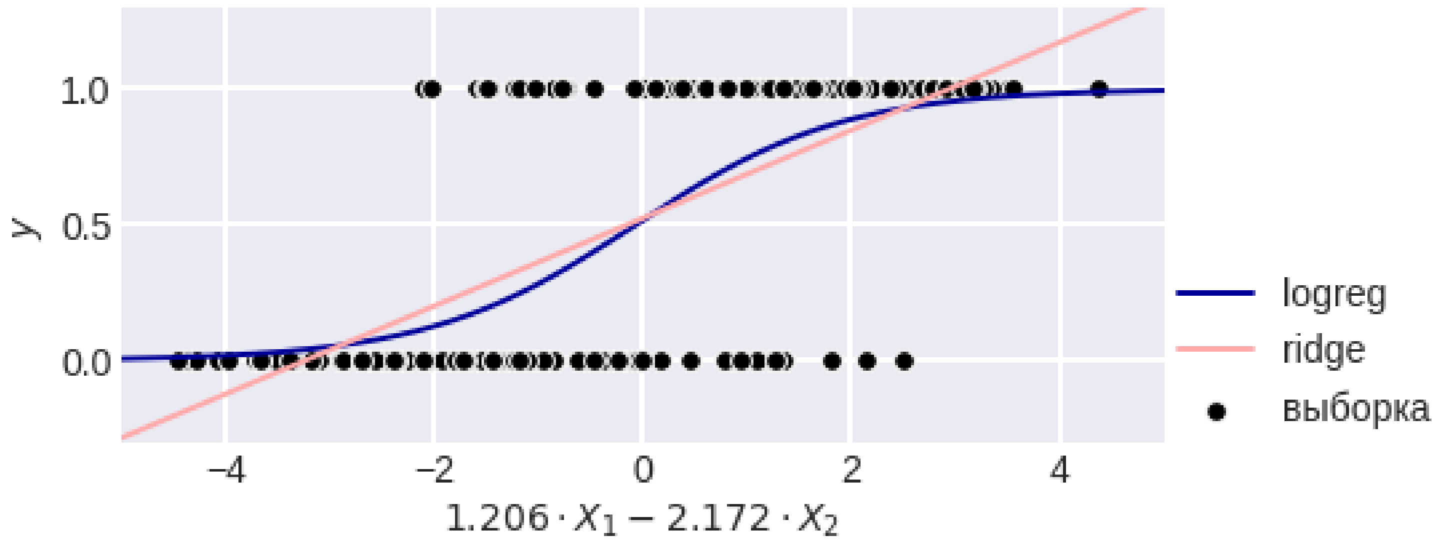




$z = w_0 + w_1 X_1 + \ldots + w_n X_n$ **– проекция на прямую (один признак)**

в однопризнаковом случае надо решить задачу классификации

Чем логистическая регрессия лучше регрессии



Обучение логистической регрессии

Метод максимального правдоподобия

$$L(w_0,\ldots,w_n)=\prod_{i:y_i=1}\sigma(w^{\mathrm{T}}x_i)\prod_{i:y_i=0}(1-\sigma(w^{\mathrm{T}}x_i))\to\max$$

здесь тоже выпуклая задача оптимизации

логирифмируем....

$$\log L(w_0,\ldots,w_n)=\sum_{i:y_i=1}\log(\sigma(w^{\mathrm{T}}x_i))\sum_{i:y_i=0}\log(\sigma(-w^{\mathrm{T}}x_i))\to\max$$

для удобства записи $y_i'=2y_i-1$, **тогда**

$$\log L=\sum_i\log(\sigma(y_i'w^{\mathrm{T}}x_i))=-\sum_i\log(1+\exp(-y_i'w^{\mathrm{T}}x_i))\to\max$$

Обучение логистической регрессии

$$\log L = -\sum_i \log(1+\exp(-y_i' w^{\mathrm{T}} x_i)) \to \max$$

Полученное выражение называют «логистической функцией ошибки» (logistic loss)

Вычислим градиент

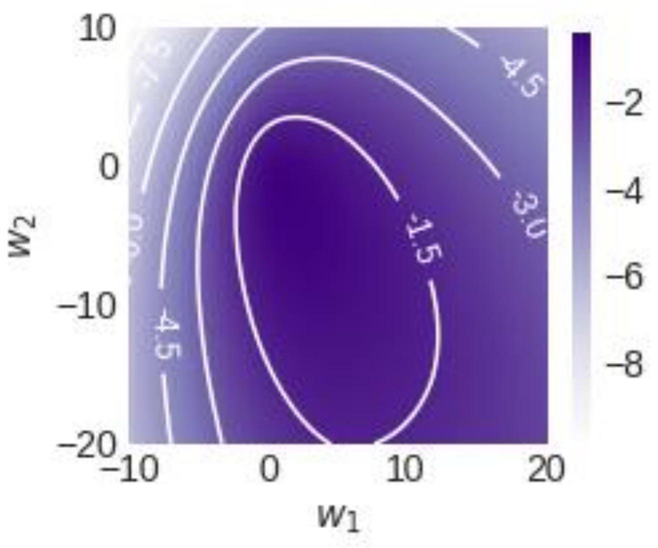
$$\nabla_w \log L = -\sum_i \nabla_w \log(1+\exp(-y_i' w^{\mathrm{T}} x_i)) =$$

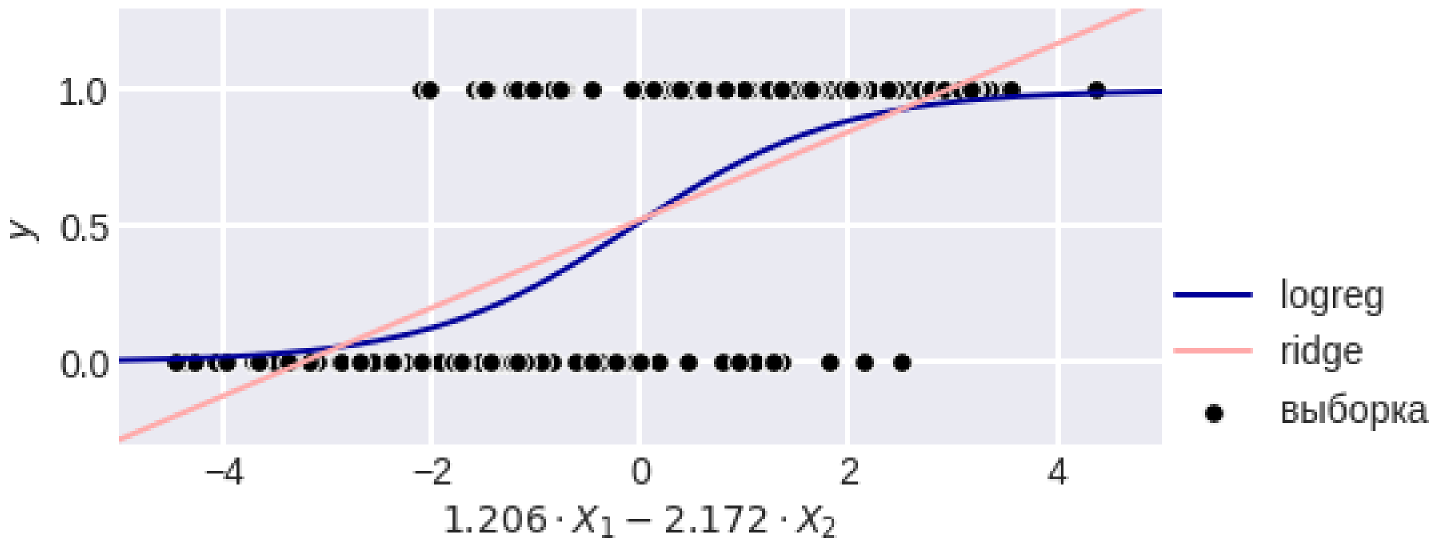
$$= -\sum_i \frac{\exp(-y_i' w^{\mathrm{T}} x_i)}{1+\exp(-y_i' w^{\mathrm{T}} x_i)}(-y_i' x_i) = \sum_i \frac{y_i' x_i}{1+\exp(+y_i' w^{\mathrm{T}} x_i)} = \sum_i \sigma(-y_i' w^{\mathrm{T}} x_i) y_i' x_i$$

Качество логистической регрессии

– логарифм правдоподобия

(потом будет соответствующая функция ошибки logloss)





метод SGD (запомним):

$$w \leftarrow w + \eta\sigma(-y_i' w^{\mathrm{T}} x_i) y_i' x_i$$

Многоклассовая логистическая регрессия: Multiclass logreg / multinomial regression

Для j-го класса имеем свою функцию $w_j^{\mathrm{T}} x$

хотим такие значения превращать в распределения

$$(w_1^{\mathrm{T}} x, w_2^{\mathrm{T}} x, \ldots, w_l^{\mathrm{T}} x) \in \mathbb{R}^l \to (p_1(x), p_2(x), \ldots, p_l(x)) \in [0,1]^l_{\Sigma=1}$$

в glmnet такой «симметричный вариант»:

$$P(Y = k \mid x) = \frac{\exp(w_{0k} + w_{1k} X_1 + \ldots + w_{nk} X_n)}{\sum_{j=1}^{l} \exp(w_{0j} + w_{1j} X_1 + \ldots + w_{nj} X_n)}$$

Такая функция называется softmax:

$$\mathrm{softmax}(a_1, \ldots, a_l) = \frac{1}{\exp(a_1) + \ldots + \exp(a_l)} [\exp(a_1), \ldots, \exp(a_l)]$$

Реализация в scikit-learn

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(penalty='l2',
                         dual=False,
                         tol=0.0001,
                         C=1.0,
                         fit_intercept=True,
                         intercept_scaling=1,
                         class_weight=None,
                         random_state=None,
                         solver='lbfgs',
                         max_iter=100,
                         multi_class='auto',
                         verbose=0,
                         warm_start=False,
                         n_jobs=None,
                         l1_ratio=None) # если penalty='elasticnet'
clf.fit(X, y)
clf.predict_proba(X)
```

Приложения: банковский скоринг

Name	Description	Type
TCS_CUSTOMER_ID	Идентификатор клиента	ID
BUREAU_CD	Код бюро, из которого получен счет	numeric
BKI_REQUEST_DATE	Дата, в которую был сделан запрос в бюро	date
CURRENCY	Валюта договора (ISO буквенный код валюты)	string
RELATIONSHIP	Тип отношения к договору	string
	1 - Физическое лицо	
	2 - Дополнительная карта/Авторизованный пользователь	
	4 - Совместный	
	5 - Поручитель	
	9 - Юридическое лицо	
OPEN_DATE	Дата открытия договора	date
FINAL_PMT_DATE	Дата финального платежа (плановая)	date
TYPE	Код типа договора	string
	1 – Кредит на автомобиль	
	4 – Лизинг. Срочные платежи за наем/пользование транспортным средством, предприятием или оборудованием и т.п.	
	6 – Ипотека – ссудные счета, имеющие отношение к домам, квартирам и прочей недвижимости. Ссуда выплачивается циклично согласно договоренности до тех пор, пока она не будет полностью выплачена или возобновлена.	
	7 – Кредитная карта	
	9 – Потребительский кредит	
	10 – Кредит на развитие бизнеса	
	11 – Кредит на пополнение оборотных средств	
	12 – Кредит на покупку оборудования	
	13 – Кредит на строительство недвижимости	
	14 – Кредит на покупку акций (например, маржинальное кредитование)	
	99 – Другой	
PMT_STRING_84M	Дисциплина (своевременность) платежей. Строка составляется из кодов состояний счета на моменты передачи банком данных по счету в бюро, первый символ - состояние на дату PMT_STRING_START, далее последовательно в порядке убывания дат.	string
	0 – Новый, оценка невозможна	
	X – Нет информации	
	1 – Оплата без просрочек	
	A – Просрочка от 1 до 29 дней	

Банковский скоринг

По описанию и истории клиента → вероятность (оценка) возврата кредита

Нужна логистическая регрессия

есть возможность получать вещественное число в виде ответа

есть более мощные методы (на решающих деревьях),

но здесь полезна интерпретация

Все категориальные признаки – OHE-перекодировка

Банковский скоринг

Если решение сводится к

$$a(x) = 1/(1+\exp(-(w_0 + w_1 X_1 + \ldots + w_n X_n))$$

где все признаки бинарные, то мы составляем скоринговую карту

Показатель	Значение показателя	Скоринг-балл
Возраст	До 30 лет От 30 до 50 лет Старше 50 лет	0 35 28
Образование	Среднее Среднее специальное Высшее	0 29 35
Состоит ли в браке	Да Нет	25 0
Брал ли кредит ранее	Да Нет	41 0
Трудовой стаж	Менее 1 года От 1 до 5 лет От 5 до 10 лет Более 10 лет	0 19 24 31

https://wiki.loginom.ru/articles/scorecard.html

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

Логистическая регрессия – деформирование линейной
Есть вероятностная трактовка!

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

«Scikit-Learn: тонкие вопросы о реализации методов машинного обучения»
https://dyakonov.org/2021/03/04/ml-scikit-learn/