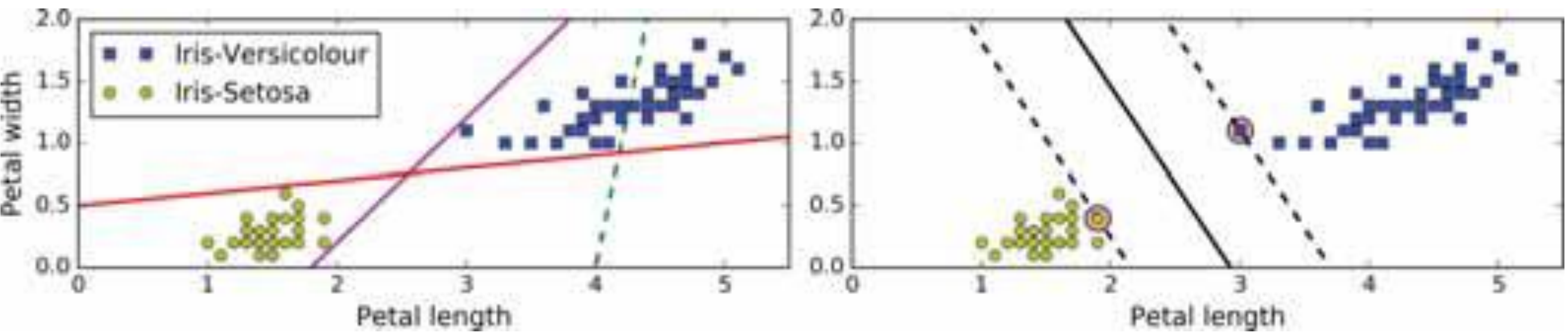
Методы опорных векторов SVM

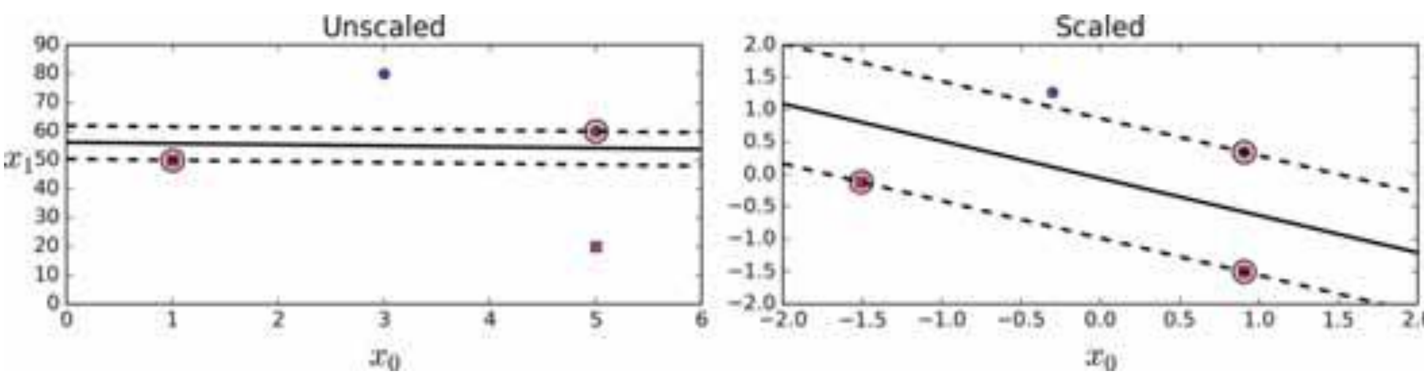
Метод опорных векторов (***Support Vector Machine*** — *SVM*) — это очень мощная и универсальная модель машинного обучения, способная выполнять линейную или нелинейную классификацию, регрессию и даже выявление выбросов. Она является одной из самых популярных моделей в МО, и любой интересующийся МО обязан иметь ее в своем инструментальном комплекте.

Методы SVM особенно хорошо подходят для классификации сложных, но небольших или средних наборов данных.

Линейная классификация SVM

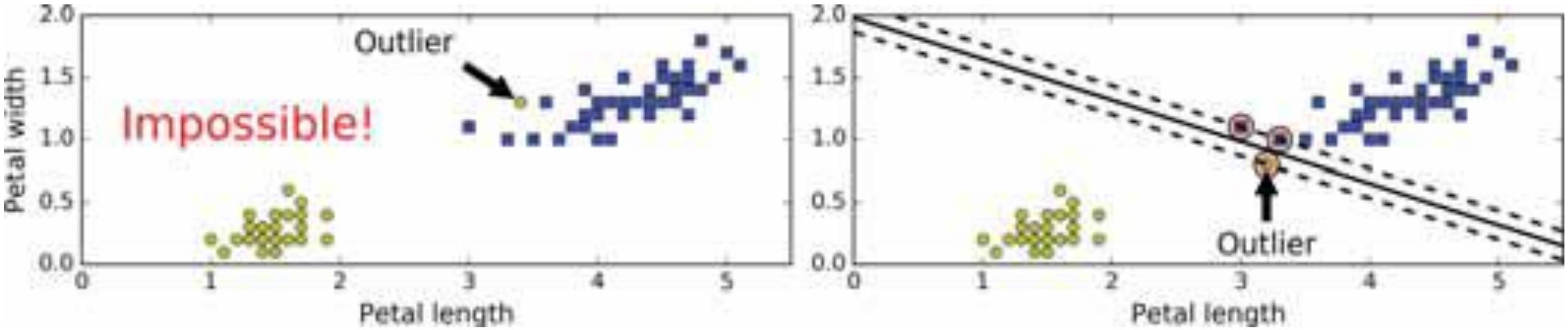
Классификатор SVM устанавливает самую широкую, какую только возможно, полосу (представленную параллельными пунктирными линиями) между классами. Это называется классификацией с широким зазором





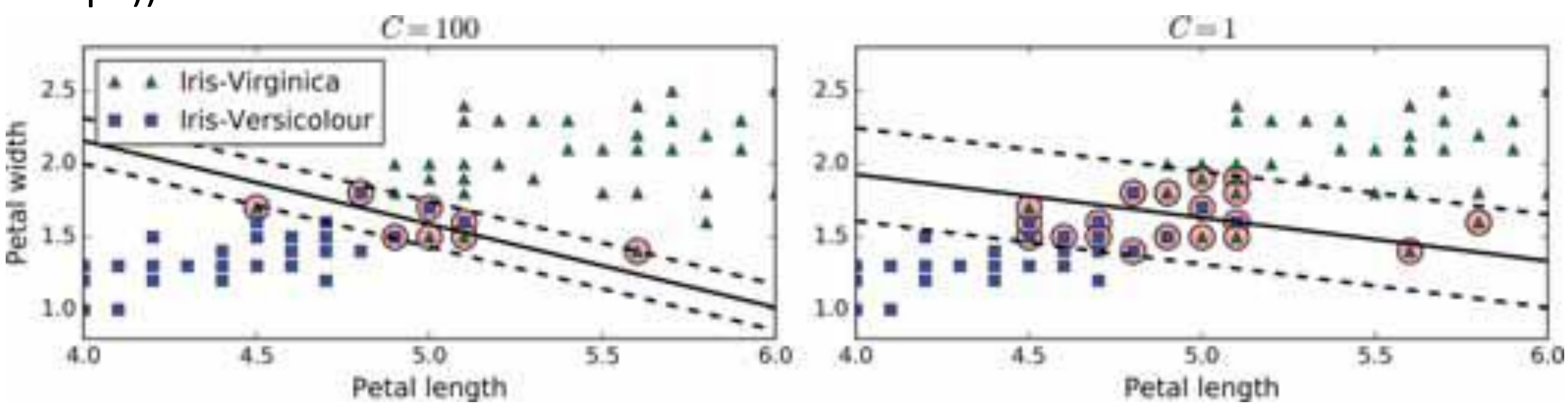
Классификация с мягким зазором

Проблема метода с жесткими зазорами



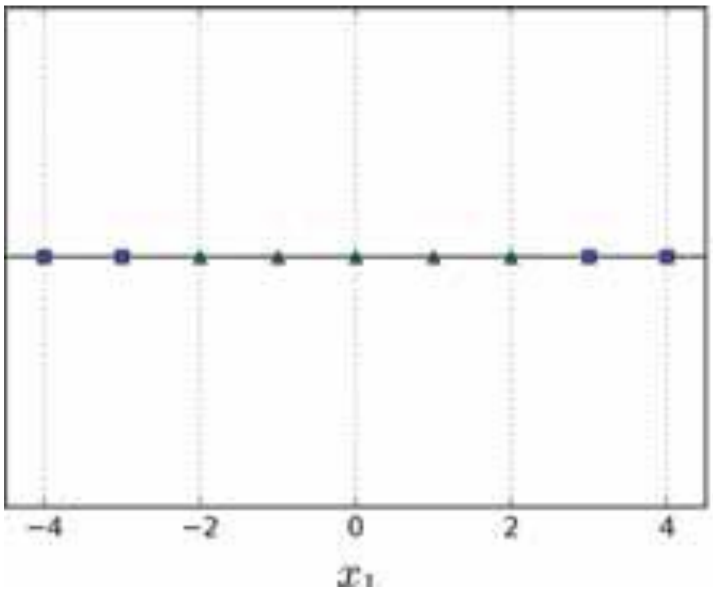
Цель заключается в том, чтобы отыскать хороший баланс между удержанием полосы как можно более широкой и ограничением количества *нарушений зазора.*

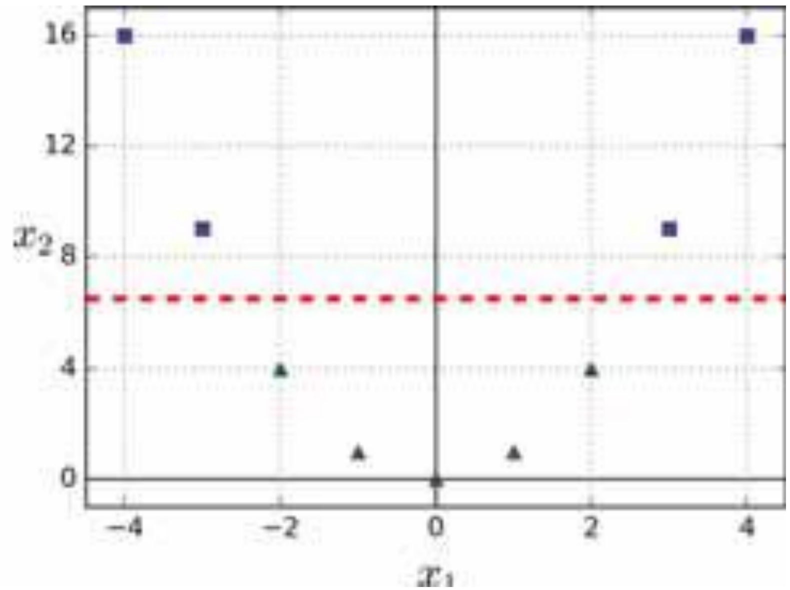
Вводится параметр штрафа С (чем выше значение С, тем меньше нарушений зазора))



Нелинейная классификация SVM

Добавление признаков для превращения набора данных в линейно разделимую $x2 = (x1)^2$



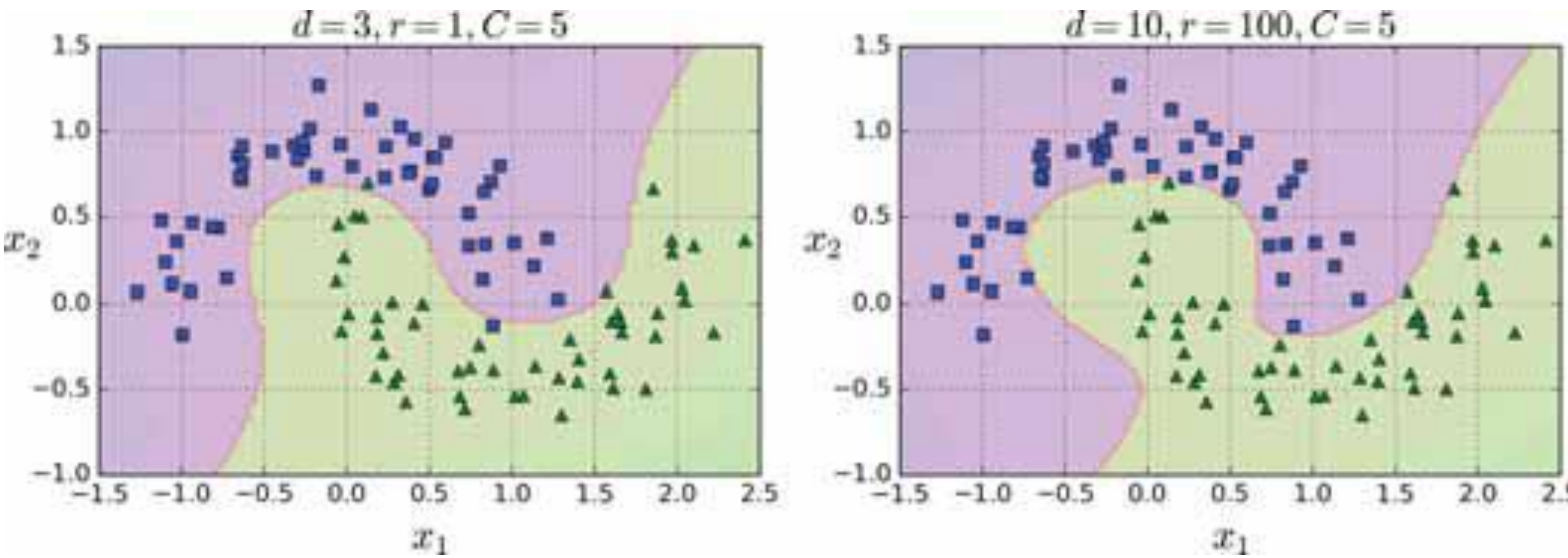


Полиномиальное ядро

Полиномиальное ядро

Добавление полиномиальных признаков просто в реализации и может великолепно работать со всеми видами алгоритмов МО

Этот код обучает классификатор SVM, использующий полиномиальное ядро 3-й степени. Он представлен слева. Справа показан еще один классификатор SVM, который применяет полиномиальное ядро 10-й степени. (r – параметр, характеризующий влияние полиномов низкой степени на полиномы высокой степени)

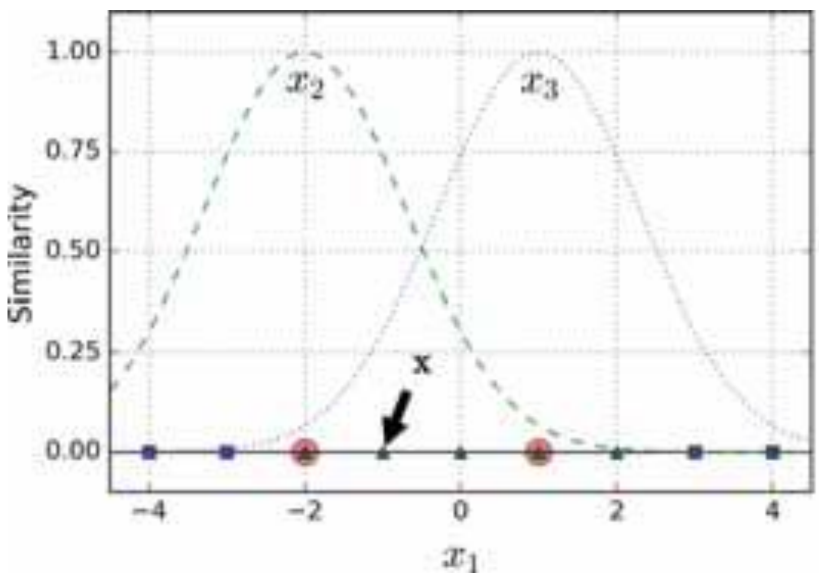


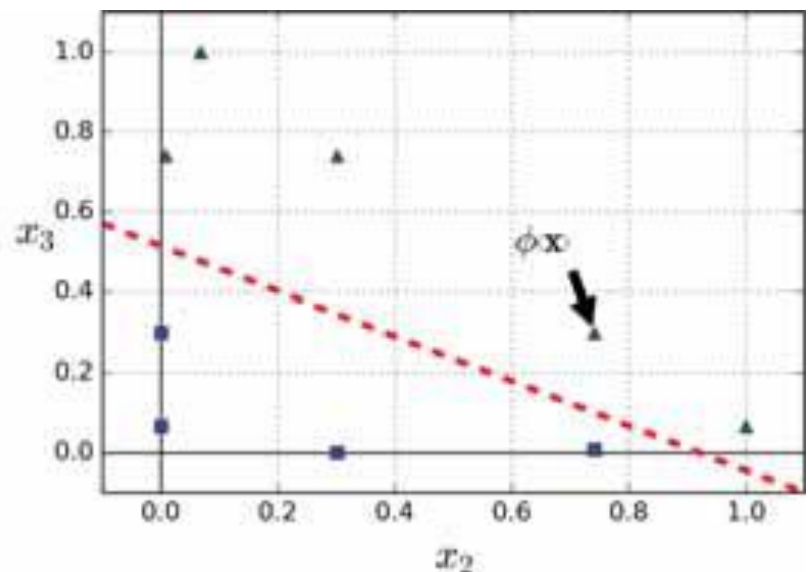
Добавление признаков близости

Возьмем обсуждаемый ранее одномерный набор данных и добавим к нему два ориентира в x1 = –2 и x1 = 1 (график слева). Затем определим функцию близости как гауссову радиальную базисную функцию (Radial Basis Function — RBF) с γ = 0.3

$$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \| \mathbf{x} - \ell \|^2\right)$$

Это колоколообразная функция, изменяющаяся от 0 (очень далеко от ориентира) до 1 (на ориентире). Теперь мы готовы вычислить новые признаки. Взглянем на образец x1 = –1: он находится на расстоянии 1 от первого ориентира и расстоянии 2 от второго ориентира. $x2 = \exp(-0.3 \times 1^2) \approx 0.74$ и $x3 = \exp(-0.3 \times 2^2) \approx 0.30$. Сделали так называемый **ядерный трюк**

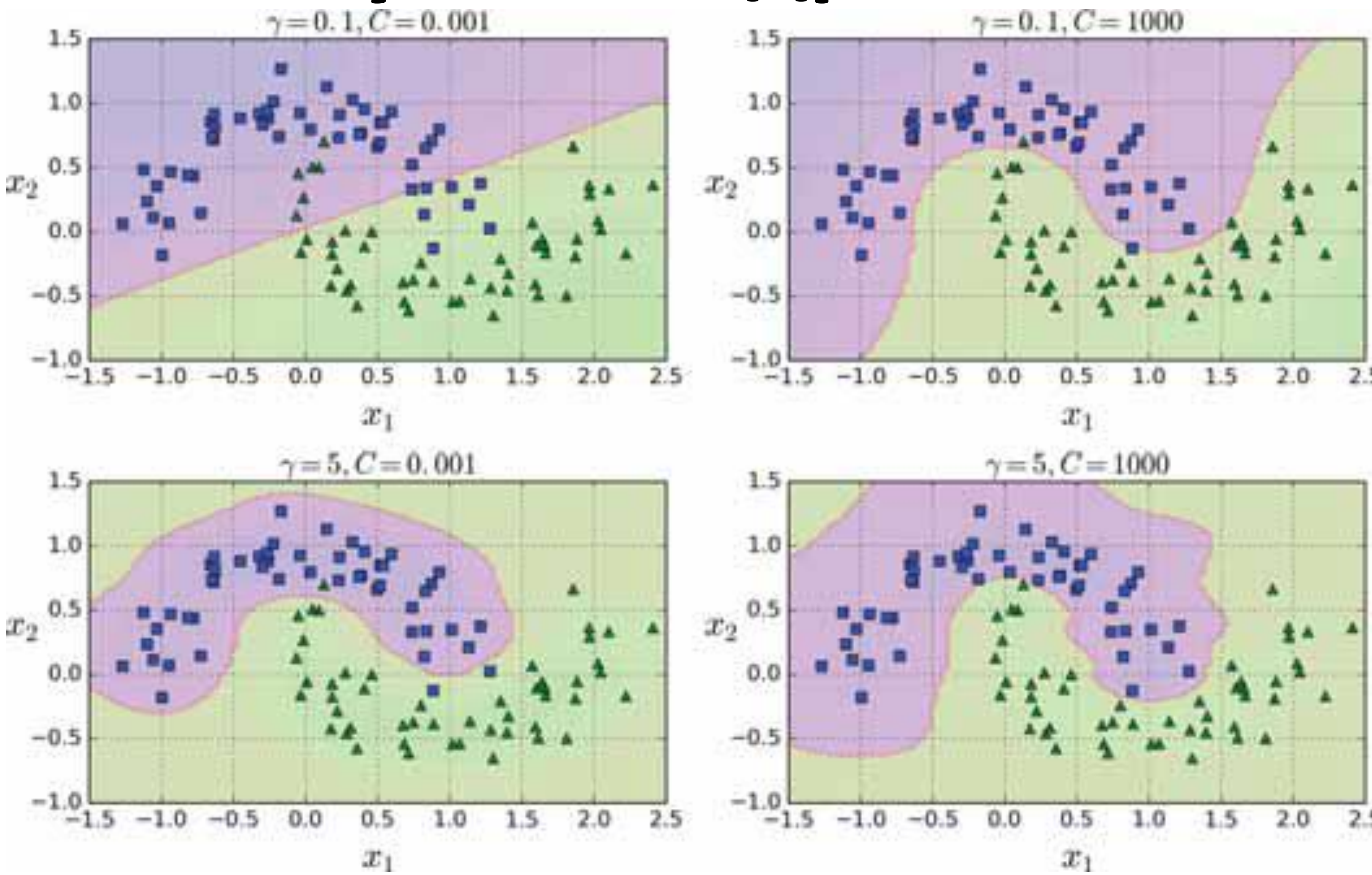




Ядерный трюк

- **Ядерный трюк** (англ. *kernel function*) метод в машинном обучении, позволяющий перевести элементы для случая линейной неразделимости в новое линейно разделимое пространство. Такое пространство называют **спрямляющим**. Поскольку для любой непротиворечивой выборки соответствующее пространство большей размерности существует, главной проблемой становится его найти.

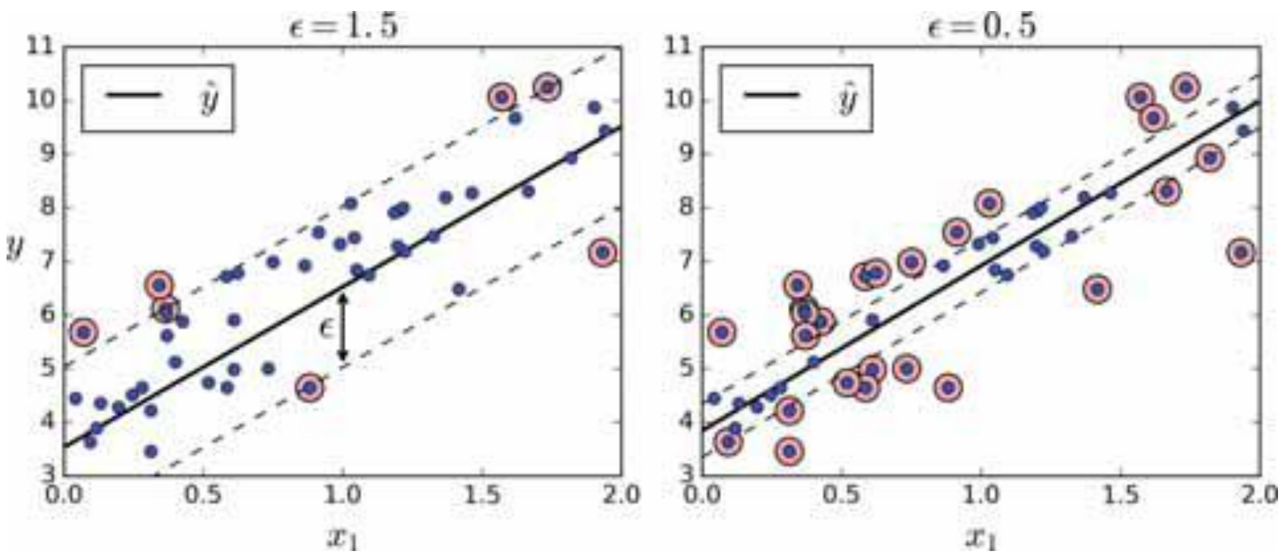
Гауссово ядро RBF



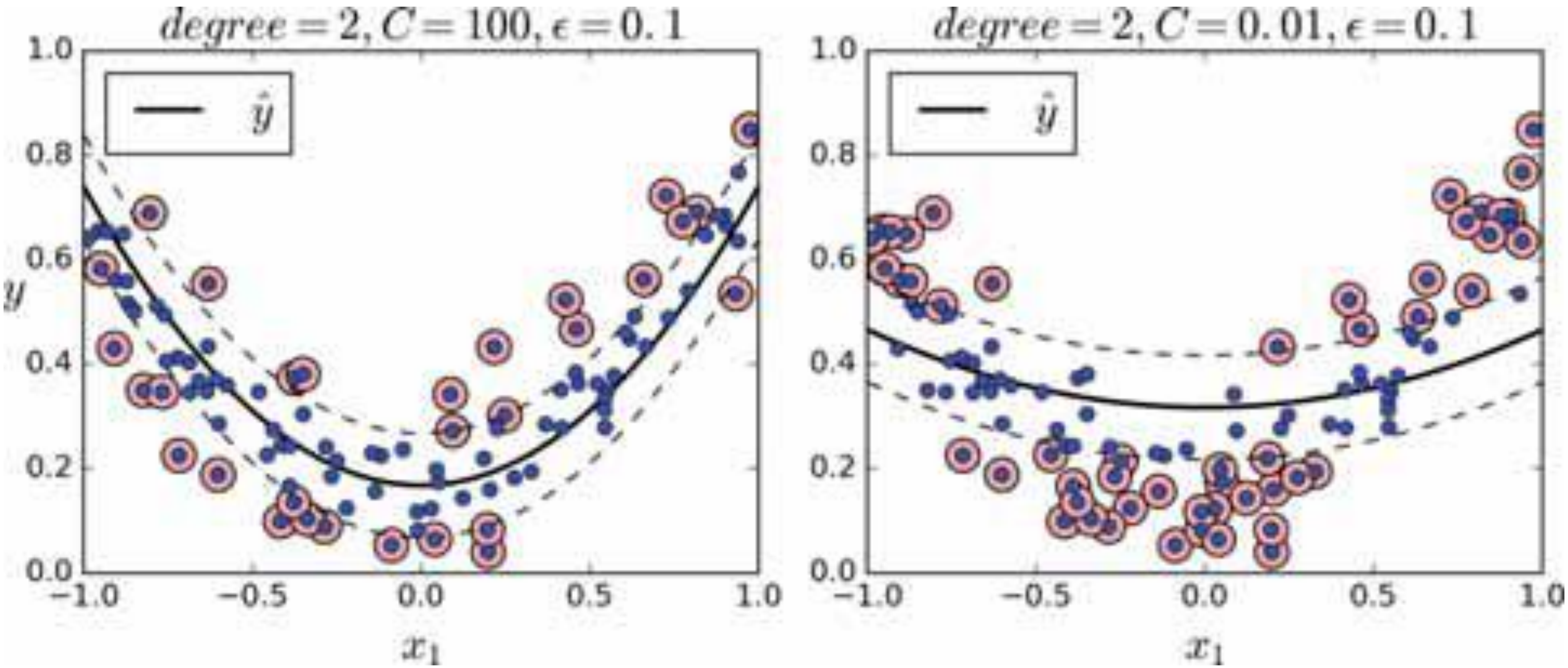
Существуют и другие ядра, но они применяются гораздо реже. Например, некоторые ядра приспособлены к специфическим структурам данных. Строковые ядра (string kernel) иногда используются при классификации текстовых документов или цепочек ДНК (например, с применением ядра строковых подпоследовательностей (string subsequence kernel).

Регрессия SVM

SVM довольно универсален: он поддерживает не только линейную и нелинейную классификацию, но также линейную и нелинейную регрессию. Прием заключается в инвертировании цели: вместо попытки приспособиться к самой широкой из возможных полосе между двумя классами, одновременно ограничивая нарушения зазора, регрессия SVM пробует уместить как можно больше образцов на полосе на ряду с ограничением нарушений зазора (т.е. образцов вне полосы). Ширина полосы управляется гиперпараметром ε. На рис. показаны две модели линейной регрессии SVM, обученные на случайных линейных данных, одна с широким зазором (ε = 1.5) и одна с узким зазором (ε = 0.5).



Регрессия SVM, применяющая полиномиальное ядро 2-го порядка



Математика

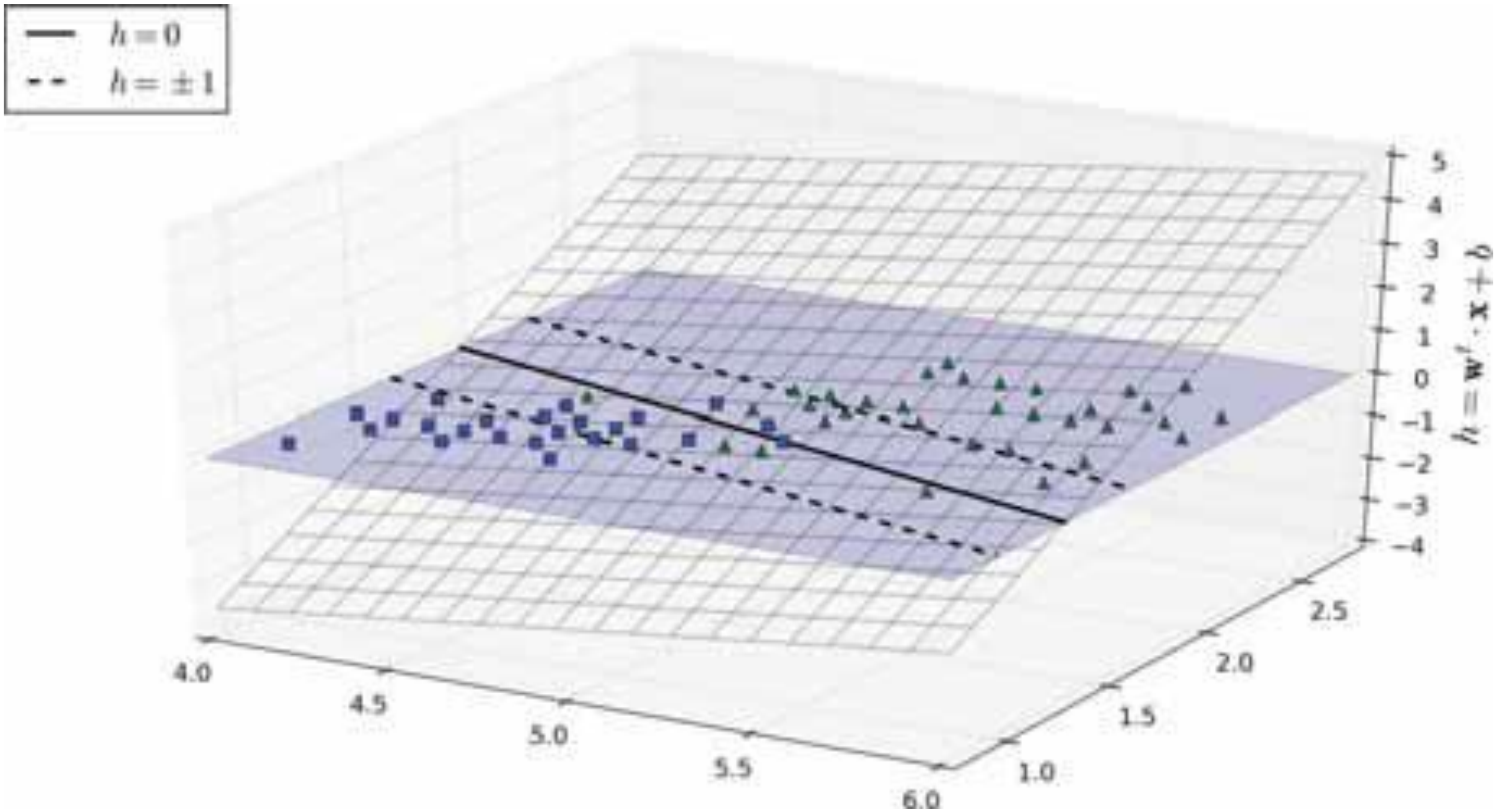
Функция решения и прогнозы

Модель линейной классификации SVM прогнозирует класс нового образца **x**, просто вычисляя функцию решения $\mathbf{w}^T * \mathbf{x} + b = w_1x_1 + \cdots + w_nx_n + b$: если результат положительный, то спрогнозированный класс является положительным (1), а иначе — отрицательным (0);

Прогноз линейного классификатора SVM

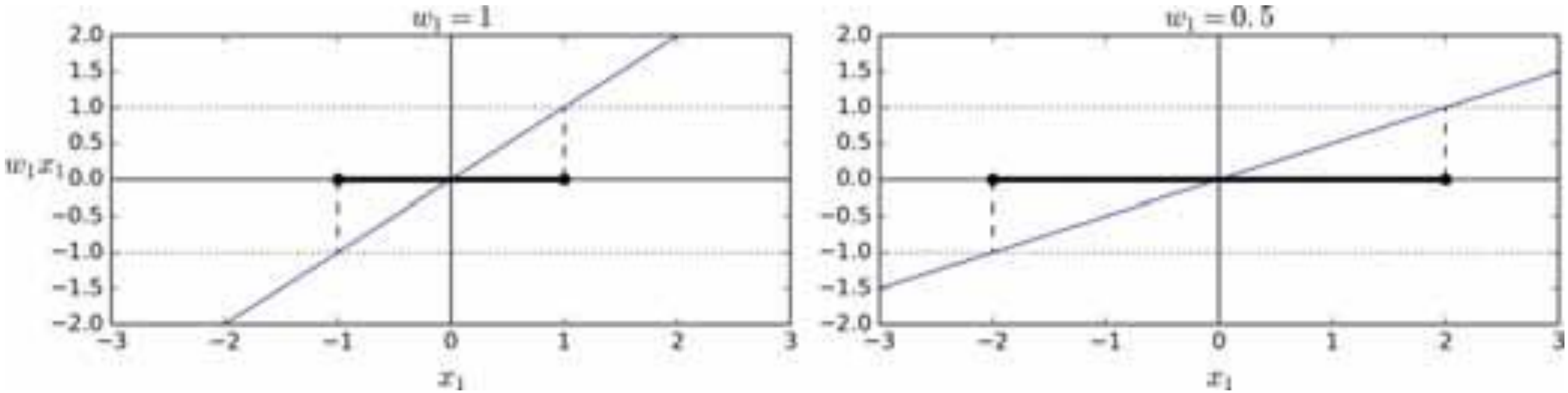
$$\hat{y} = \begin{cases} 0, & \text{если } \mathbf{w}^T \cdot \mathbf{x} + b < 0, \\ 1, & \text{если } \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \end{cases}$$

Функция решения для набора данных об ирисах



Цель обучения

Рассмотрим наклон функции решения: он тождественен норме вектора весов, $\|w\|$. Если мы разделим наклон на 2, тогда точки, в которых функция решения равна ±1, будут в два раза дальше от границы решений. Другими словами, деление наклона на 2 умножит зазор на 2.



Меньший вектор весов приводит к более широкому зазору

Цель обучения

если мы также хотим избежать любых нарушений зазора (иметь жесткий зазор), то нужно, чтобы функция решения была больше 1 для всех положительных обучающих образцов и меньше –1 для отрицательных обучающих образцов. Если мы определим $t^{(i)} = -1$ для отрицательных образцов (когда $y^{(i)} = 0$) и $t^{(i)} = 1$ для положительных образцов (когда $y^{(i)} = 1$), тогда можем выразить такое ограничение как $t^{(i)} (\mathbf{w}^T * \mathbf{x}^{(i)} + b) \geq 1$ для всех образцов.
Следовательно, мы можем выразить цель линейного классификатора SVM с жестким зазором как задачу *условной оптимизации*

Цель линейного классификатора SVM с жестким зазором

$$\underset{\mathbf{w},\, b}{\text{минимизировать}} \quad \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}$$

$$\text{при условии} \quad t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) \geq 1 \quad \text{для} \quad i = 1, 2, \ldots, m$$

Мы минимизируем ½ w^T*w , что равносильно ½ $\|w\|^2$, вместо минимизации $\|w\|$. Причина в том, что это даст тот же самый результат (поскольку значения **w** и *b*, которые минимизируют какое-то значение, также минимизируют половину его квадрата), но ½ $\|w\|^2$ имеет подходящую и простую производную (именно **w**), в то время как $\|w\|$ не дифференцируется для **w** = 0. Алгоритмы оптимизации гораздо лучше работают с дифференцируемыми функциями.

Цель линейного классификатора SVM с мягким зазором

Чтобы достичь цели мягкого зазора, нам необходимо ввести *фиктивную переменную* (***slack variable***) $\zeta^{(i)} \geq 0$ для каждого образца: $\zeta^{(i)}$ измеряет, на сколько i-тому образцу разрешено нарушать зазор. Теперь мы имеем две противоречивые цели: делать фиктивные переменные как можно меньшими, чтобы сократить нарушения зазора, и делать ½ w^T*w как можно меньшим, чтобы расширить зазор. Именно здесь в игру вступает гиперпараметр (*параметр штрафа*) C: он позволяет нам определить компромисс между указанными двумя целями. Мы получаем задачу условной оптимизации

Цель линейного классификатора SVM с мягким зазором

$$\underset{\mathbf{w}, b, \zeta}{\text{минимизировать}} \quad \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C \sum_{i=1}^{m} \zeta^{(i)}$$

$$\text{при условии} \quad t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) \geq 1 - \zeta^{(i)} \quad \text{и} \quad \zeta^{(i)} \geq 0 \quad \text{для} \quad i = 1, 2, \dots, m$$

Двойственная задача (обратная)

Имея задачу условной оптимизации, известную как *прямая задача* (***primal problem***), можно выразить отличающуюся, но тесно связанную задачу, которая называется *двойственной задачей* (***dual problem***).

Двойственная форма цели линейного классификатора SVM

$$\underset{\alpha}{\text{минимизировать}} \;\; \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha^{(i)}\alpha^{(j)} t^{(i)} t^{(j)} \mathbf{x}^{(i)T} \cdot \mathbf{x}^{(j)} - \sum_{i=1}^{m}\alpha^{(i)}$$

$$\text{при условии} \quad \alpha^{(i)} \geq 0 \;\; \text{для} \;\; i = 1, 2, \ldots, m$$

Принцип двойственности в задачах линейного программирования.

Предположим, что руководство предприятия из анализа конъюнктуры рынка продукции приняли решение: производство сократить, а от запасов сырья избавиться, (продать на рынке) и при этом не нанести себе убытков.

С этой целью руководство должно назначить стоимости y_i за единицу сырья вида S_i, стремясь при этом минимизировать общую стоимость сырья (чтобы быстрее продать сырье): $\Phi = \Sigma^4_{i=1} b_i y_i$

Выручка предприятия от продажи сырья, расходуемого на единицу продукции Π_i, составит: $\Sigma^4_{i=1} a_{ij} y_i$

И по условию она не должна быть меньше C_j (в противном случае предприятию выгоднее не продавать сырье, а использовать его для нужд производства, выпуска продукции).

Сформулируем исходную и двойственную задачи:

		Виды задач ЛП	
Виды формулировок		Исходная	Двойственная
	Развернутая	Найти максимум $Q = c_1x_1+c_2x_2$ При ограничениях $a_{11}x_1+a_{12}x2 \le b_1,$ $a_{21}x_1+a_{22}x2 \le b_2,$ $a_{31}x_1+a_{32}x2 \le b_3,$ $a_{41}x_1+a_{42}x2 \le b_4,$ $x_i \ge 0,$	Найти минимум $Q' = b_1y_1+ b_2y_2+ b_3y_3+ b_4y_4$ При ограничениях $a_{11}y_1+ a_{21}y_2+ a_{31}y_3+ a_{41}y_4 \ge c_1$ $a_{12}y_1+ a_{22}y_2+ a_{32}y_3+ a_{42}y_4 \ge c_2$ $y_i \ge 0$ $Y_{<4>}$ - вектор цен видов сырья
	Векторно-матричная	$X_{<2>}$ - вектор объемов продукции Найти максимум $Q = C^T_{<2>} X_{<2>},$ при $A_{[42]} X_{<2>} \le B_{<4>},$ $X_{<2>} \ge O_{<4>}.$	Найти минимум $Q' = B^T_{<4>} Y_{<2>},$ $A^T_{[4,2]} = A_{[2,4]} Y_{<4>} \ge C_{<2>},$ $Y_{<4>} \ge O_{<4>}.$

После нахождения вектора $\hat{\alpha}$, сводящего к минимуму это уравнение (используя решатель QP), с применением уравнения вы можете вычислить $\hat{\mathbf{w}}$ и $\hat{b}$, которые минимизируют прямую задачу.

От решения двойственной задачи к решению прямой задачи

$$\hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} \mathbf{x}^{(i)}$$

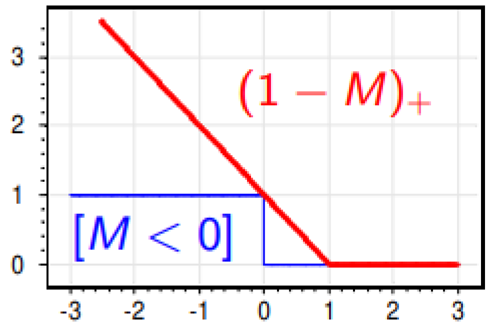
$$\hat{b} = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)} > 0}}^{m} \left(t^{(i)} - \hat{\mathbf{w}}^T \cdot \mathbf{x}^{(i)} \right)$$

Двойственная задача решается быстрее прямой, когда количество обучающих образцов меньше количества признаков. Что более важно, становится возможным ядерный трюк, в то время как при решении прямой задачи он невозможен. Итак, что же собой представляет этот самый ядерный трюк?

Функция потерь для SVM

кусочно-линейная (SVM) *(hinge loss)*

$$(1 - M)_+$$




```
svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge")),
    ])
```