Тверской
государственный
технический
университет

Интеллектуальные информационные системы

Метод опорных векторов
Решающие деревья

2025 г.

Метод опорных векторов в задачах классификации

Задача бинарной классификации: $X = \mathbb{R}^n$,

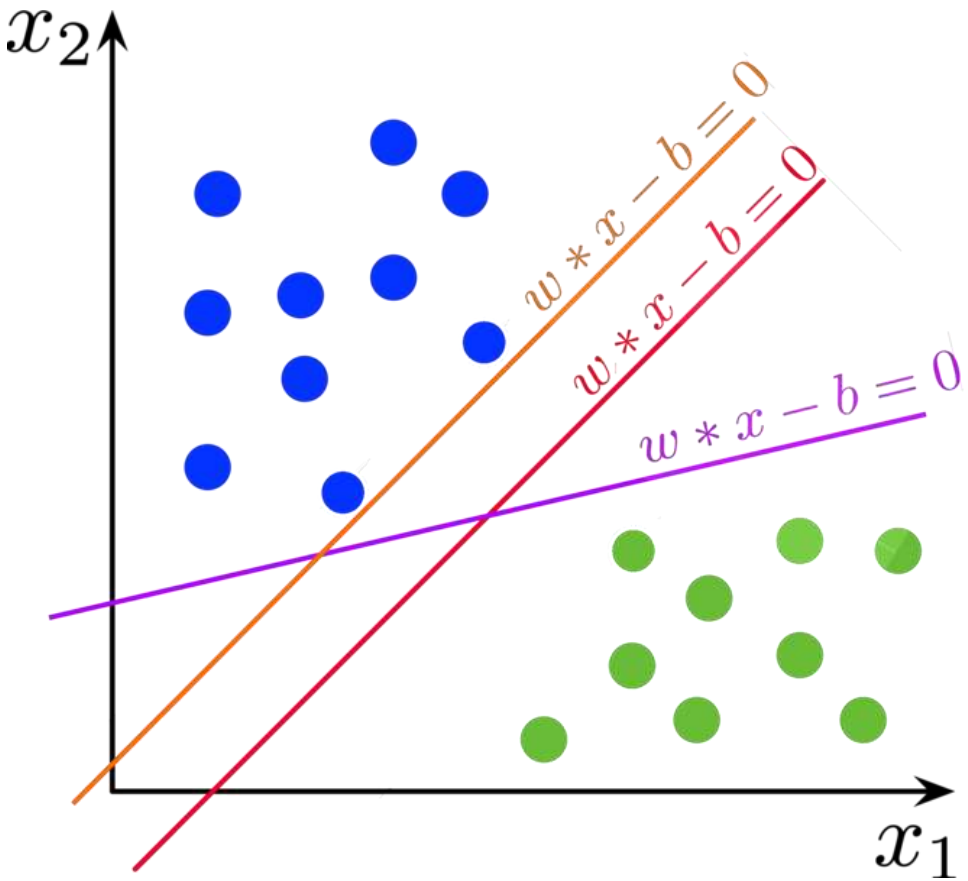
$$Y = \{-1, +1\}$$

Линейный классификатор:

$$a(x) = sign\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = sign(\langle w, x\rangle - w_0)$$

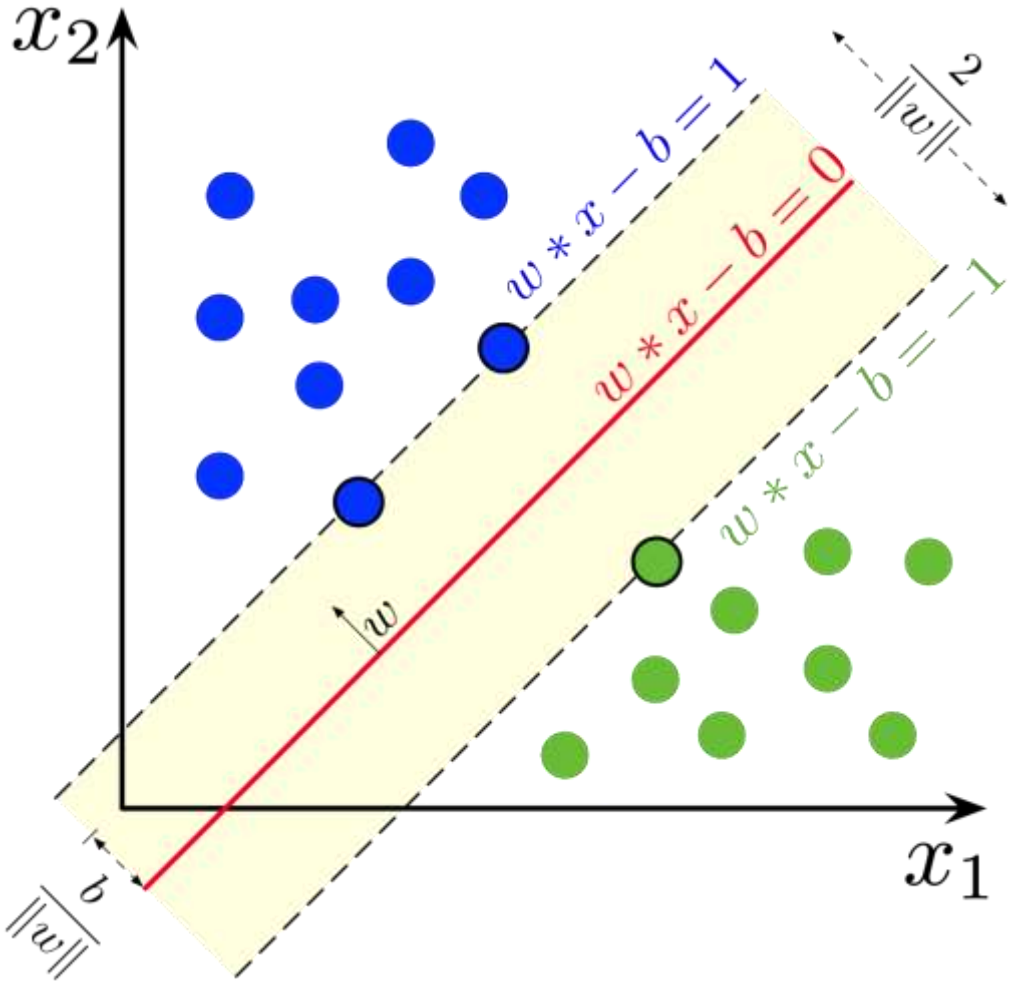
Предположим, что выборка линейно разделима:

$$Q(w, w_0) = \sum_{i=1}^{l} [y_i(\langle w, x_i \rangle - w_0) < 0] = 0$$



Ширина полосы:

$$\left\langle (\boldsymbol{x}_+ - \boldsymbol{x}_-), \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right\rangle = \frac{\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_- \rangle}{\|\boldsymbol{w}\|} = \frac{(\boldsymbol{w_0} + \mathbf{1}) - (\boldsymbol{w_0} - \mathbf{1})}{\|\boldsymbol{w}\|} = \frac{\mathbf{2}}{\|\boldsymbol{w}\|}$$



Для линейно разделимой выборки:

Классификация с жестким зазором (hard margin)

$$\begin{cases} \frac{1}{2}\|w\|^2 \to \min\limits_{w, w_0} \\ M_i(w, w_0) \geq 1, i = 1, \ldots, l \end{cases}$$

Задача квадратичного программирования - минимизировать квадратичный функционал при линейных ограничениях

Обобщение для линейно неразделимой выборки:

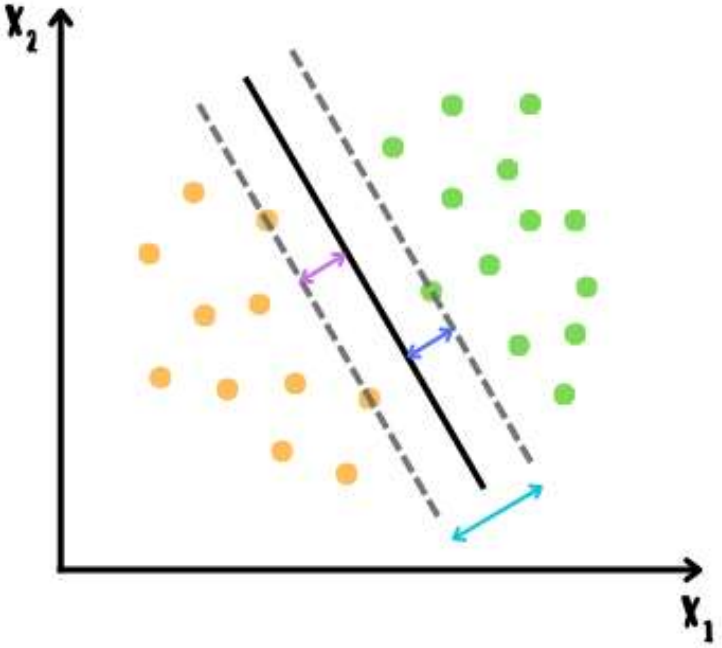
Классификация с мягким зазором (soft margin)

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{l} \xi_i \to \min\limits_{w, w_0, \xi} \\ M_i(w, w_0) \geq 1 - \xi_i, \quad i = 1, \ldots, l \\ \xi_i \geq 0, \quad i = 1, \ldots, l \end{cases}$$

ξ_i – величина ошибки на объектах x_i

C – штраф за суммарную ошибку

hard margin



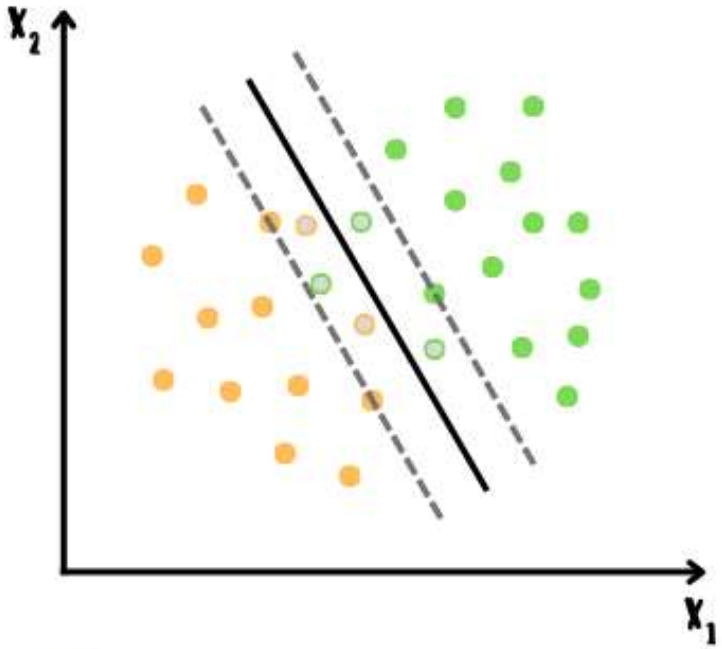
margin

margin

total margin

support vectors

soft margin

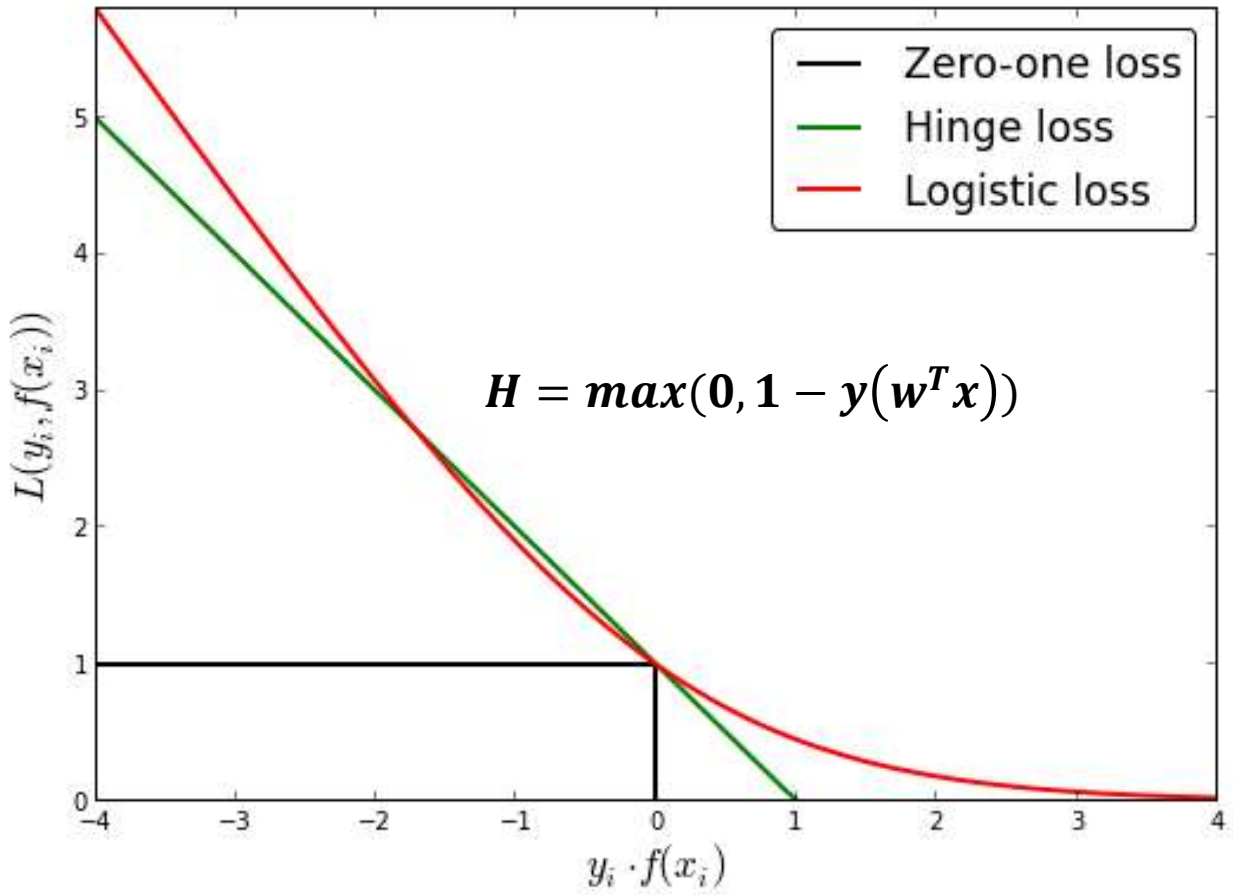


samples outside the support vectors

samples outside the support vectors and the hyperplane

Эквивалентная задача безусловной оптимизации:

$$C\sum_{i=1}^{l}(1-M_i(w,w_0))_+ + \frac{1}{2}\|w\|^2 \to \min_{w,w_0}$$



Инструмент для решения задачи оптимизации с ограничениями равенства и неравенства –

условия Каруша-Куна-Таккера

$$\begin{cases} f(x) \to min \\ g_i(x) \le 0, \quad i = 1, \dots, m \\ h_j(x) = 0, \quad j = 1, \dots, k \end{cases}$$

По теореме ККТ эта задача эквивалентна двойственной задаче поиска седловой точки функции Лагранжа

Если x – точка локального минимума, то существуют множители $\mu_i, i = 1, \dots, m$ и $\lambda_i, j = 1, \dots, k$:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial x} = 0, \qquad \mathcal{L}(x; \mu, \lambda) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{j=1}^{k} \lambda_j h_j(x) \\ g_i(x) \le 0;\ h_j(x) = 0 \\ \mu_i \ge 0 \\ \mu_i \vee g_i(x) = 0 \end{cases}$$

Функция Лагранжа: $\mathcal{L}(w, w_0, \xi; \lambda, \eta) =$

$$\frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \lambda_i (M_i(w, w_0) - 1) - \sum_{i=1}^{l} \xi_i (\lambda_i + \eta_i - C)$$

Продифференцируем функцию Лагранжа и приравняем нулю производные:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow \boxed{w = \sum_{i=1}^{l} \lambda_i y_i x_i}$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = 0 \Rightarrow \boxed{\sum_{i=1}^{l} \lambda_i y_i = 0}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow \boxed{\eta_i + \lambda_i = C,} \qquad i = 1, \dots, l$$

Вектор весов $\boldsymbol{w}$ выражается через $\boldsymbol{\lambda_i}$:

$$\boldsymbol{w} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{l}} \boldsymbol{\lambda_i y_i x_i}$$

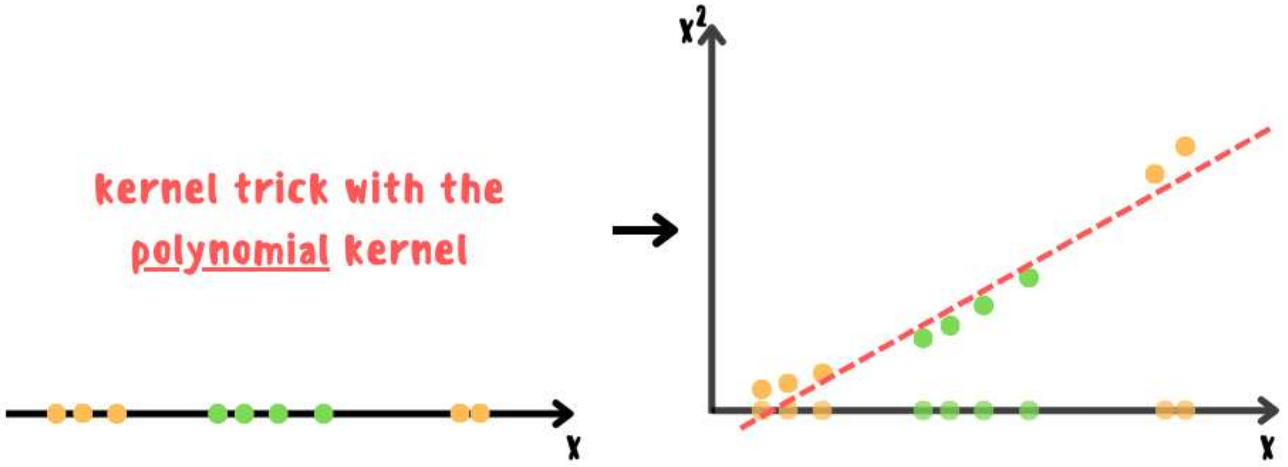
Если $\boldsymbol{\lambda_i = 0}$, то решение задачи никак не зависит от объекта обучающей выборки $\boldsymbol{x_i}$ (эти объекты можно назвать неинформативными)

Объект $\boldsymbol{x_i}$ называется *опорным*, если $\boldsymbol{\lambda_i \neq 0}$

Нелинейное обобщение SVM

Идея: переход от исходного пространства признаковых описаний объектов X к новому пространству H (*спрямляющему* пространству) с помощью некоторого преобразования $\boldsymbol{\psi}: \boldsymbol{X} \rightarrow \boldsymbol{H}$

Kernel trick (ядерный трюк) – замена скалярного произведения векторов n-й степени заменяется на их произведение в степени n: $\boldsymbol{\psi(a)^T \psi(b) = \left(a^T b\right)^n}$



- Линейное ядро:

$$K(x, x^T) = \langle x, x^T \rangle$$

- Квадратичное ядро:

$$K(x, x^T) = \langle x, x^T \rangle^2$$

- Полиномиальное ядро:

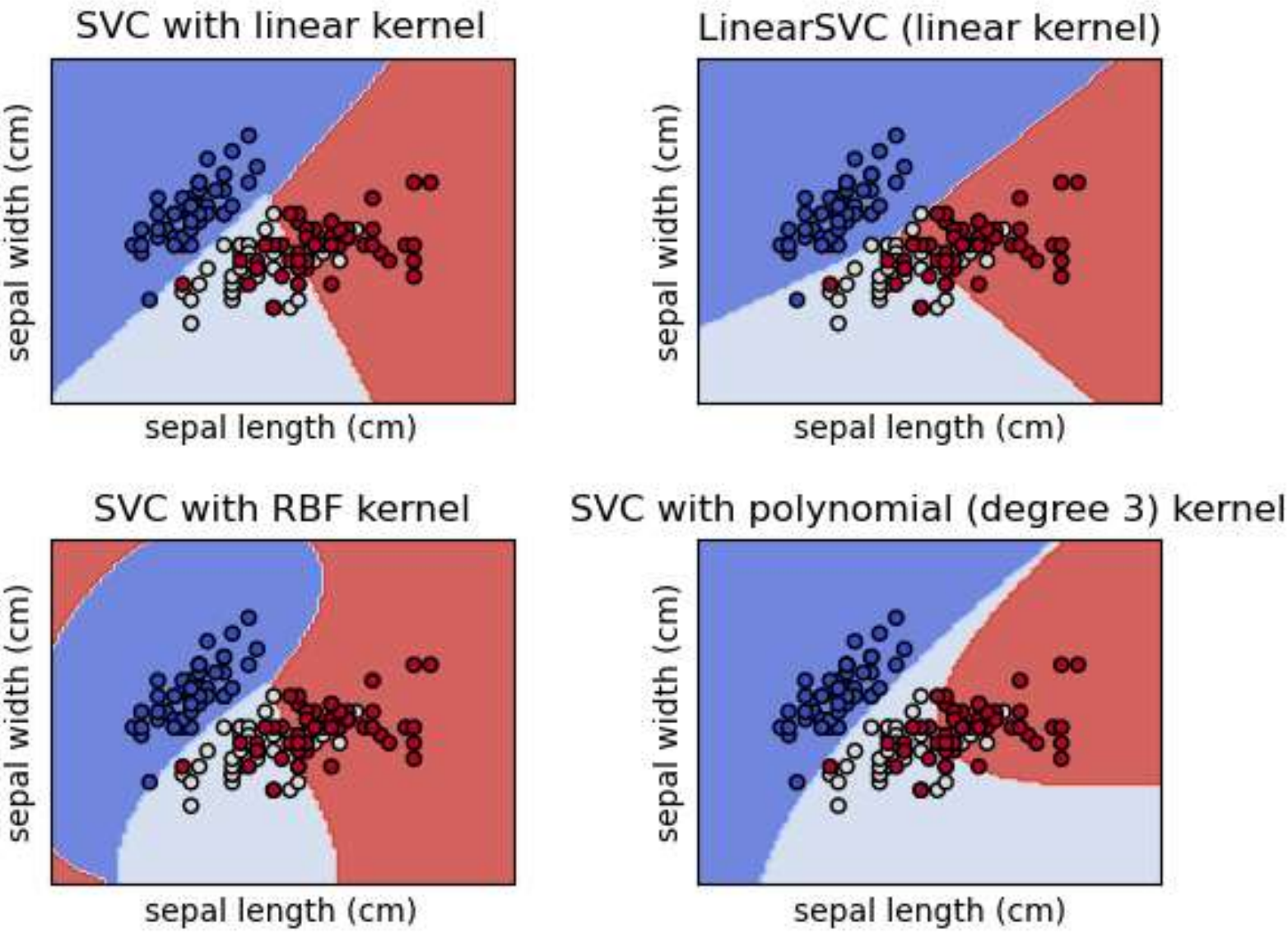
$$K(x, x^T) = (\langle x, x^T \rangle + 1)^d$$

- Гауссовское RBF ядро (радиально-базисная функция):

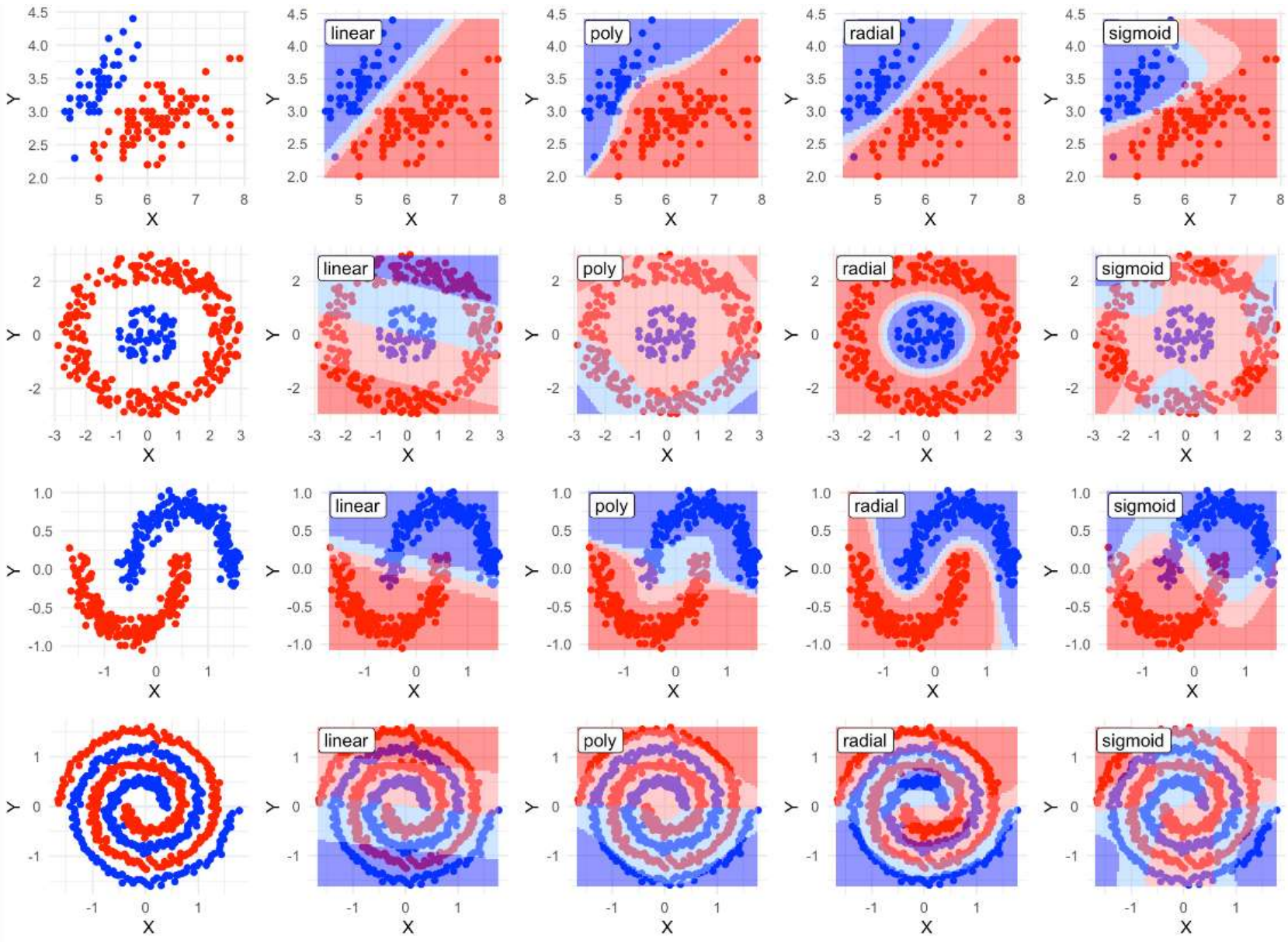
$$K(x, x^T) = \exp(-\gamma \|x - x^T\|^2)$$

- Сигмоидальное ядро:

$$K(x, x^T) = \tanh(k_1 \langle x, x^T \rangle - k_0), \quad k_0, k_1 \geq 0$$



Источник: https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html



Источник: https://rpubs.com/jmsallan/SVMkernels

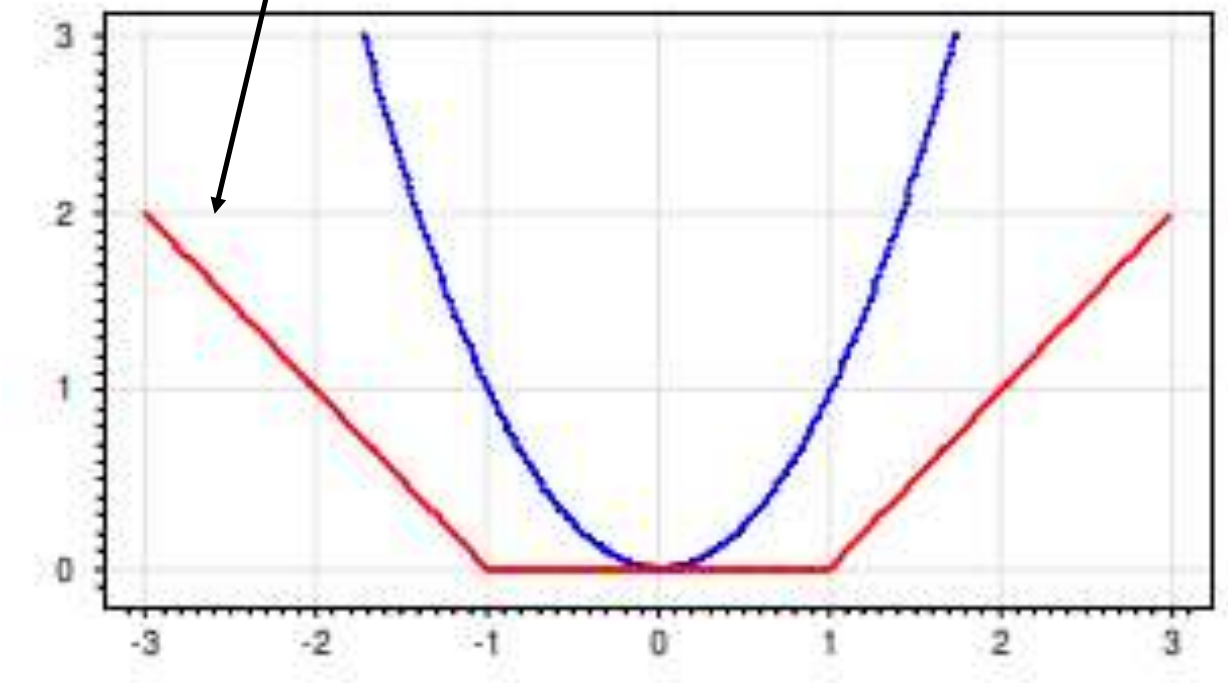
Метод опорных векторов в задачах регрессии

Модель регрессии:

$$a(x) = \langle x, w \rangle - w_0, \qquad w \epsilon \mathbb{R}^n, \; w_0 \epsilon \mathbb{R}$$

Функция потерь: $L(\varepsilon) = (|\varepsilon| - \delta)_+$

Кусочно-линейная функция ε-чувствительности



Смысл функции потерь – *если мы находимся на расстоянии не более чем* $\boldsymbol{\delta}$ *от правильного ответа, то потери нет, за такой ответ не штрафуем. Если дальше, то штраф увеличивается линейно*.

Постановка задачи (с L2-регуляризацией):

$$\sum_{i=1}^{l} (|\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle - \boldsymbol{w_0} - \boldsymbol{y_i}| - \boldsymbol{\delta})_+ + \frac{\mathbf{1}}{\boldsymbol{2C}} \|\boldsymbol{w}\|^2 \to \min_{\boldsymbol{w}, \boldsymbol{w_0}}$$

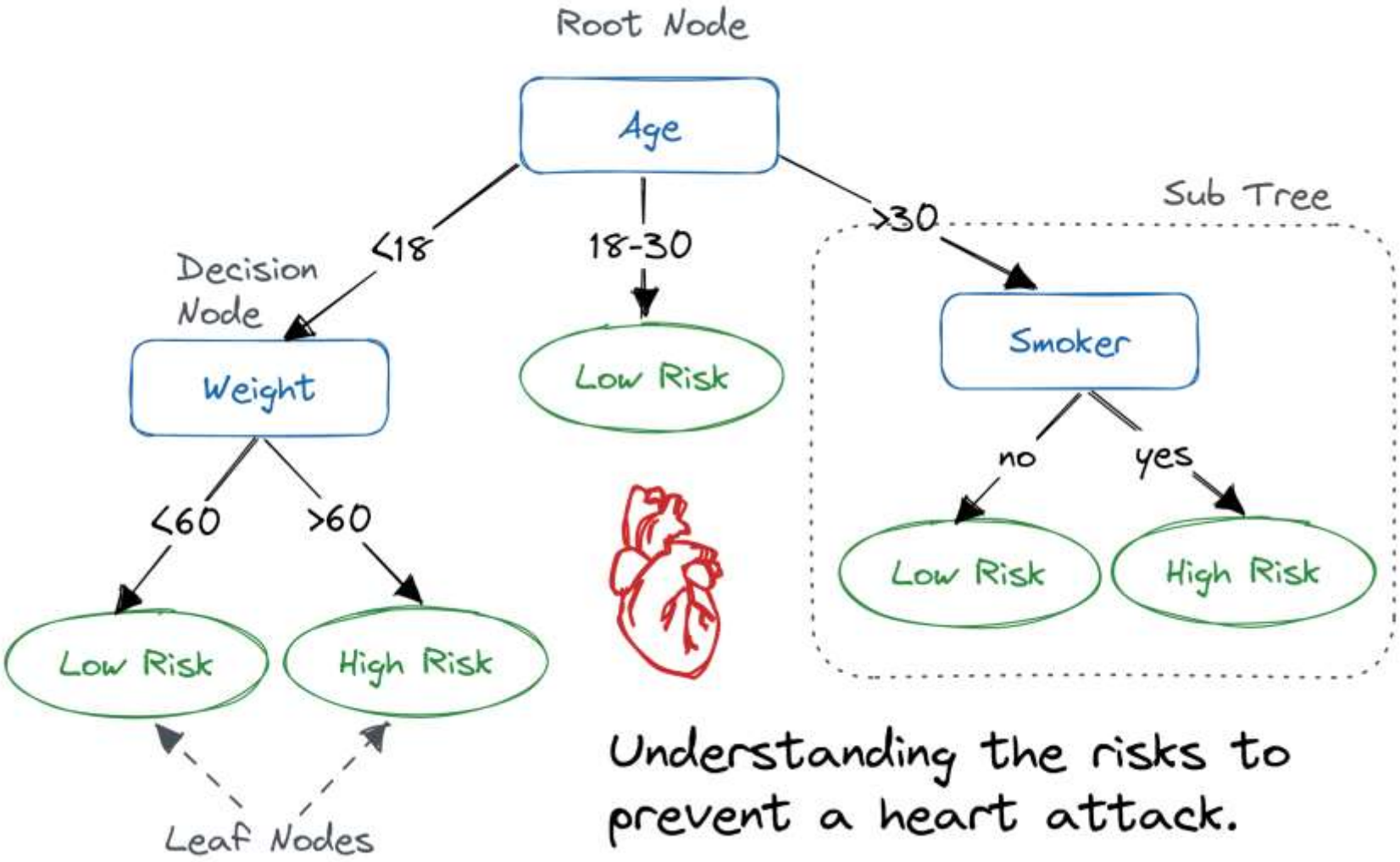
Достоинства SVM:

- ✓ Задача выпуклого квадратичного программирования имеет единственное решение – методы оптимизации существенно более эффективны
- ✓ Принцип оптимальной разделяющей гиперплоскости приводит к максимизации разделяющей полосы между классами – более уверенная классификация
- ✓ Возможность обработки многомерных данных без предварительного преобразования

Недостатки SVM:

- × Неустойчивость к шуму, выбросы в обучающей выборке могут стать опорными объектами-нарушителями и непосредственно влиять на построение разделяющей гиперплоскости
- × Не существует общих методов построения ядер и спрямляющих пространств
- × Необходимо подбирать параметр C в случае линейной неразделимости

Решающие деревья



Источник: https://www.datacamp.com/tutorial/decision-tree-classification-python

Решающее дерево – конечный связный граф с множеством вершим V, не содержащий циклов и имеющий выделенную вершину (*корень* дерева) $\boldsymbol{v_0} \in \boldsymbol{V}$, в которую не входит ни одно ребро. Вершина, не имеющая выходящих ребер, называется терминальной или *листом*. Остальные вершины называются *внутренними*

Бинарное решающее дерево – решающее дерево, в котором каждой внутренней вершине $\boldsymbol{v} \in \boldsymbol{V}$ приписан предикат (функция) $\boldsymbol{\beta_v} : \boldsymbol{X} \rightarrow \{\mathbf{0}, \mathbf{1}\}$ и каждой терминальной вершине приписано имя класса

Предикат – логическое выражение или условие, которое используется в узле решения для выбора ветви. Могут быть как одномерные, так и многомерные.

- *Одномерные* предикаты – сравнивают значение одного из признаков с порогом:

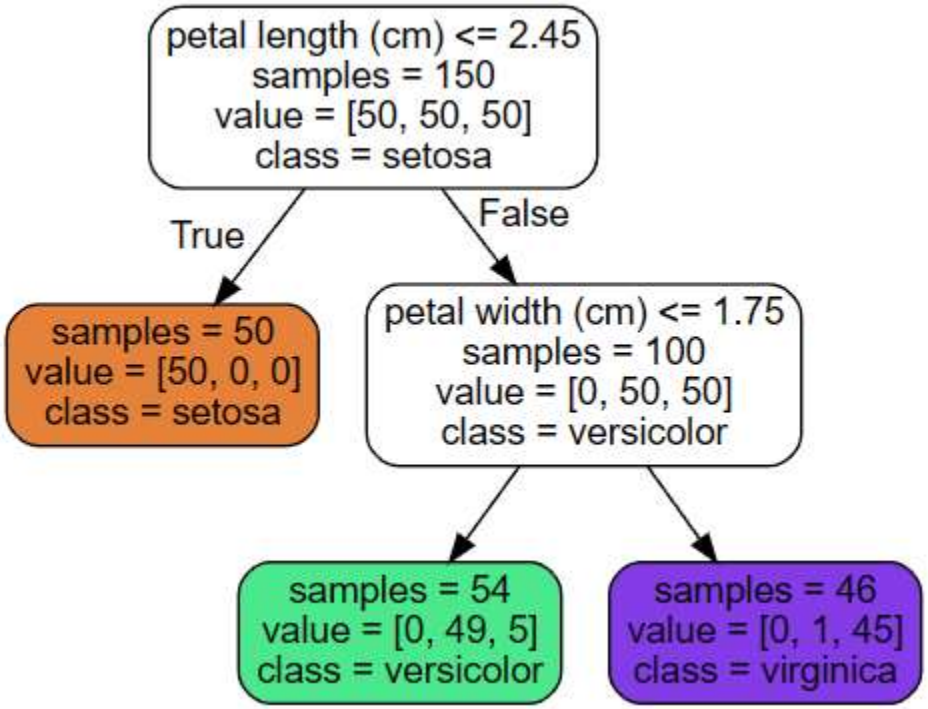
$$\boldsymbol{\beta_v(x; j, t) = [x_j < t]}$$

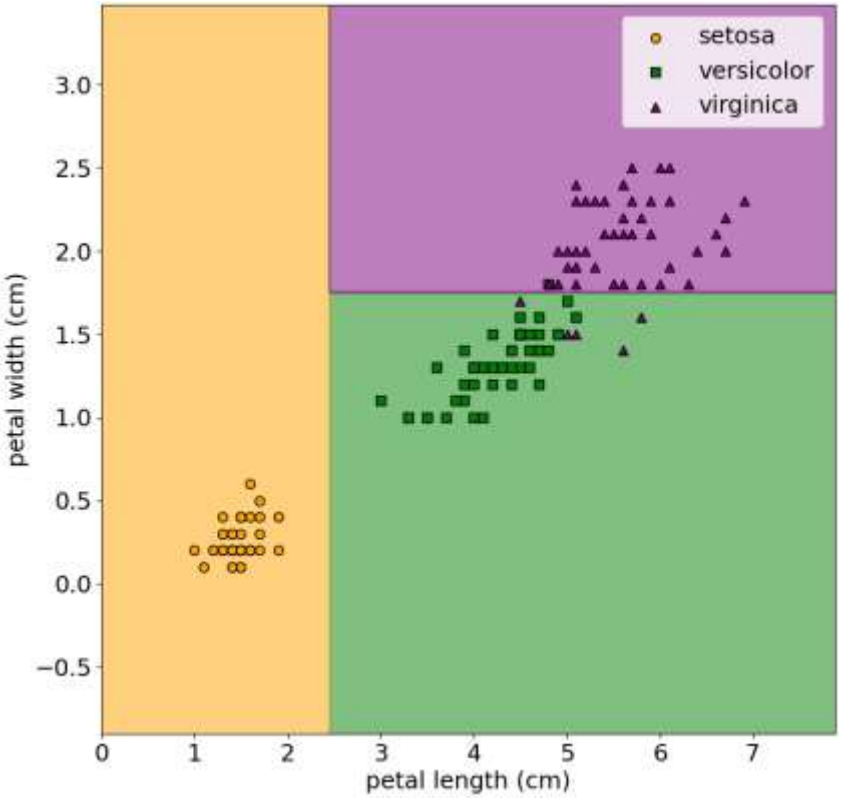
- *Многомерные* предикаты – используют несколько признаков. Например, линейный многомерный предикат записывается как:

$$\boldsymbol{\beta_v(\mathbf{x}) = [\langle \mathbf{w}, \mathbf{x} \rangle < \mathbf{t}]}$$

Decision tree classifier on Iris data

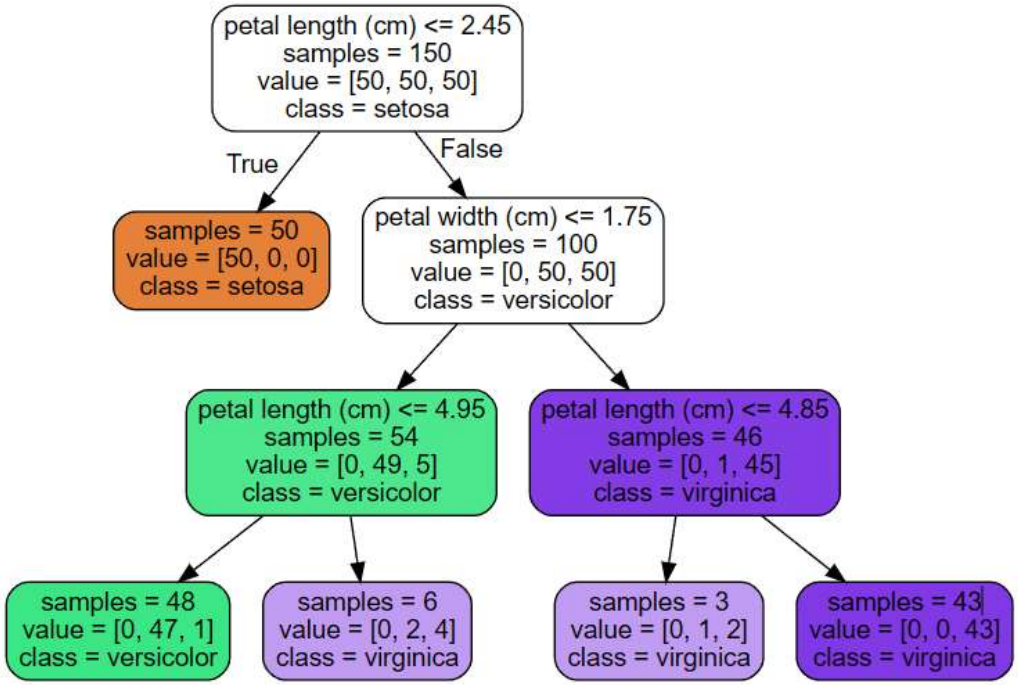
Depth = 2

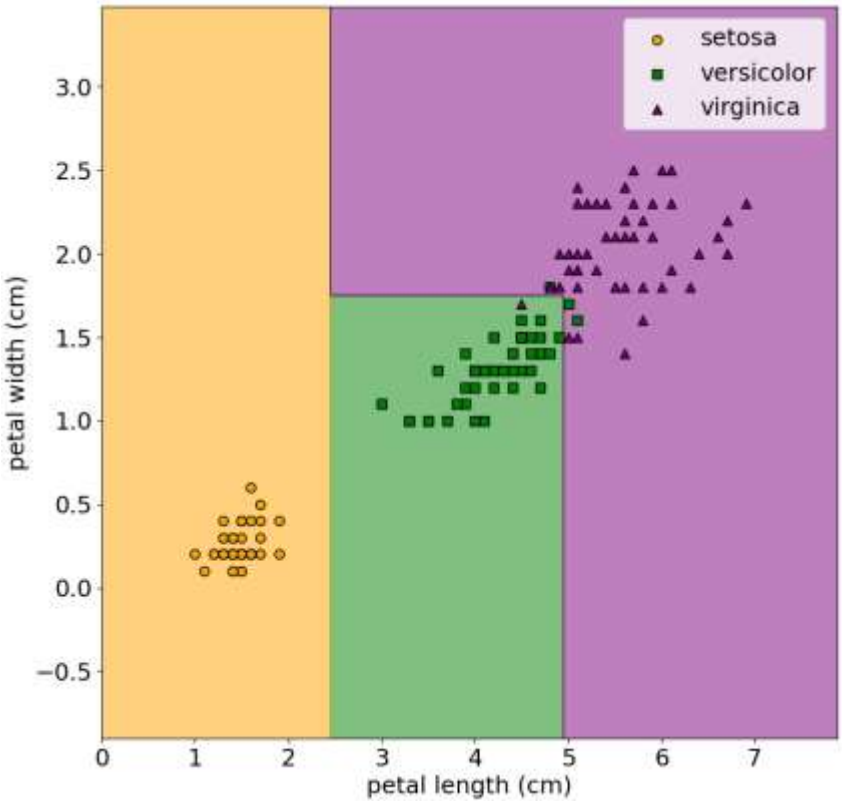




Decision tree classifier on Iris data

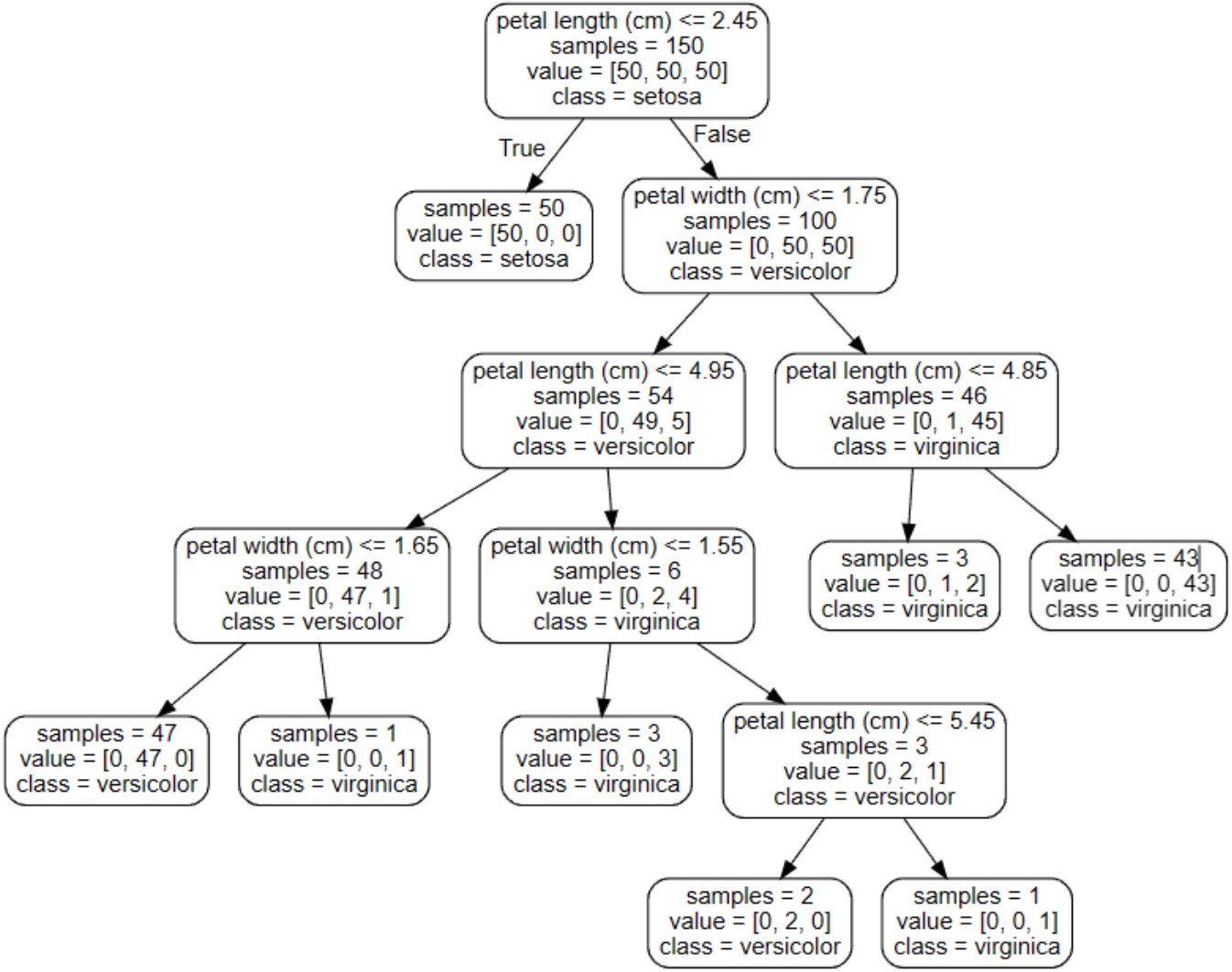
Depth = 3

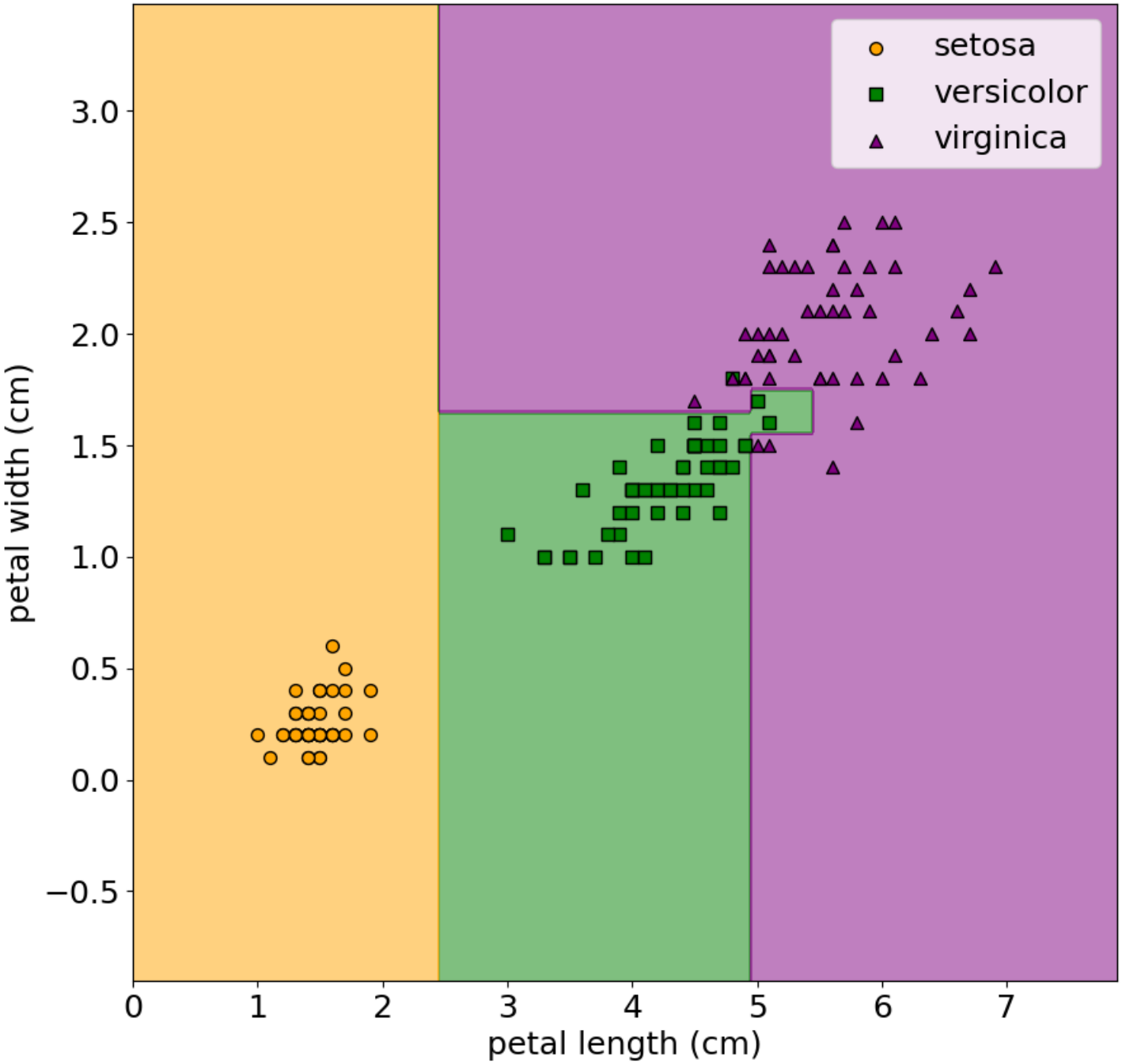


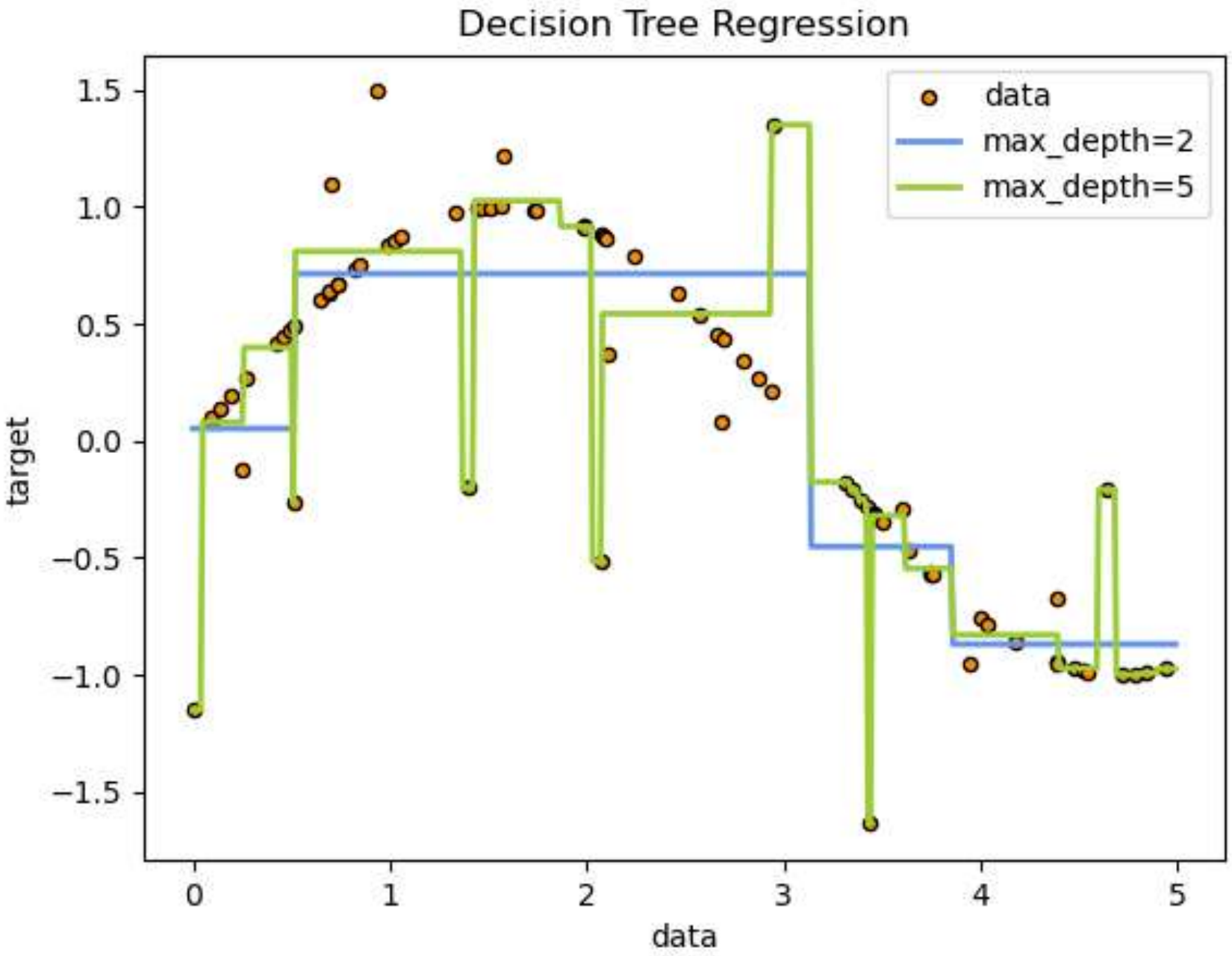


Decision tree classifier on Iris data

Depth unrestricted




setosa
versicolor
virginica
petal width (cm)
petal length (cm)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7



Источник: https://scikit-learn.org/1.5/auto_examples/tree/plot_tree_regression.html

Построение решающих деревьев

X – исходное множество объектов обучающей выборки

X_m – множество объектов, попавших в текущий лист

Жадный алгоритм:

1. Создаем вершину ***v***
2. Если выполнен *критерий остановки* $\boldsymbol{Stop(X_m)}$, то останавливаемся, объявляем эту вершину листом и ставим ей в соответствие ответ $\boldsymbol{Ans(X_m)}$ и возвращаем ее
3. Иначе: находим предикат $B_{j,t}$, который определит наилучшее разбиение текущего множества объектов X_m на две подвыборки X_l и X_r, максимизируя *критерий ветвления* $\boldsymbol{Branch(X_m, j, t)}$
4. Для X_l и X_r рекурсивно повторяем процедуру

- Критерий остановки $\boldsymbol{Stop}(\boldsymbol{X_m})$ – функция, которая решает, нужно ли продолжать ветвление

- Функция $\boldsymbol{Ans}(\boldsymbol{X_m})$ – вычисляет ответ для листа по попавшим в него объектам из обучающей выборки. Для классификации - метка самого частого класса, оценка дискретного распределения вероятностей, для регрессии – статистикой(мода, медиана, среднее)

- Критерий ветвления $\boldsymbol{Branch}(\boldsymbol{X_m}, \boldsymbol{j}, \boldsymbol{t})$ – функция, измеряющая, насколько хорош предлагаемый сплит. Чаще всего оценивает улучшение финальной метрики качества дерева в случае, если получившиеся два листа будут терминальными, в сравнении с ситуацией, когда сама исходная вершина будет листом. Выбирается сплит, который дает наиболее существенное улучшение

Критерий ветвления определяется оценкой

<u>критерия информативности (impurity)</u> :

$$H(X_m) = min \frac{1}{|X_m|} \sum_{(x_i, y_i) \in X_m} L(y_i, c)$$

c – ответы дерева, с $\in R$ для регрессии и меток класса, для дискретного распределения на классах с $\in R^K$ вектор вероятностей $\mathrm{c} = (\mathrm{c}_1, \ldots, c_K), \sum_{i=1}^{K} c_i = 1$

$L(y_i, c)$ – функция потерь

$$\boldsymbol{Branch(X_m, j, t) = |X_m| \cdot H(X_m) - |X_l| \cdot H(X_l) - |X_r| \cdot H(X_r)}$$

Чтобы принять решение о разделении, необходимо сравнить значение информативности для исходного листа и для получившегося после разделения решающего пня. Чем больше – тем лучше предполагаемый сплит

Критерии информативности в задачах регрессии

- *MSE* – регрессия с минимизацией квадратичной ошибки.

$$H(X_m) = \sum_{(x_i,y_i)\in X_m} \frac{(y_i - \overline{y})^2}{|X_m|}, \overline{y} = \frac{1}{|X_m|}\sum_i y_i$$

- *MAE* – регрессия с минимизацией абсолютной ошибки

$$H(X_m) = \sum_{(x_i,y_i)\in X_m} \frac{|y_i - MEDIAN(Y)|}{|X_m|}$$

Критерии информативности в задачах классификации

- *Misclassification error* – доля ошибок:

$$\boldsymbol{H(X_m) = \frac{1}{|X_m|} \sum_{(x_i, y_i) \in X_m} [y_i \neq k_*] = 1 - p_{k_*}}$$

k_*- наиболее частый класс, $\mathrm{p_{k_*}}$ - доля объектов класса k в текущей вершине $\mathrm{X_m}$

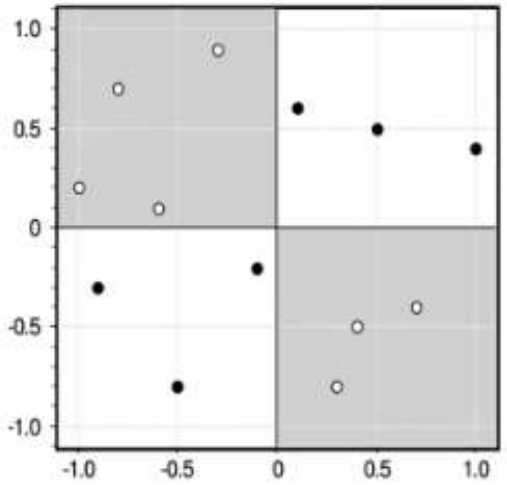
- *Энтропия* распределения классов (информационная энтропия Шеннона, измеряет непредсказуемость реализации случайной величины)

$$\boldsymbol{H(X_m) = -\sum_{k=1}^{K} p_k log p_k}$$

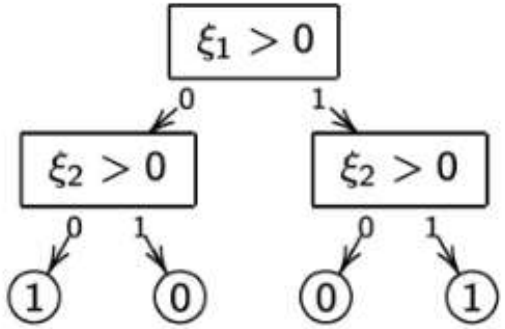
- *Критерий Джини* – $\boldsymbol{H(X_m)}$ равно математическому ожиданию числа неправильно классифицированных объектов в случае, если мы будем приписывать им случайные метки из дискретного распределения, заданного вероятностями $(p_1, \dots, p_k)$

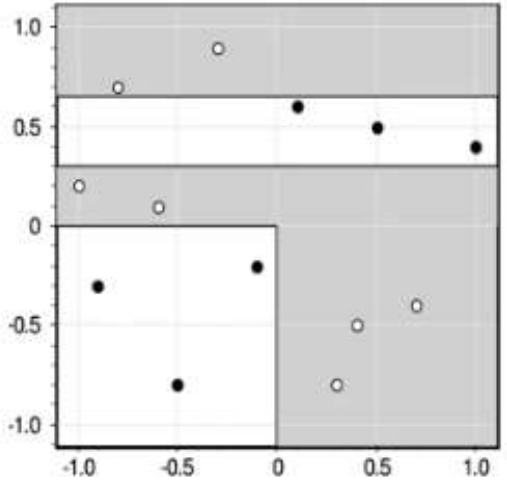
$$\boldsymbol{H(X_m) = \sum_{k=1}^{K} p_k(1 - p_k)}$$

Неоптимальность полученных критериев

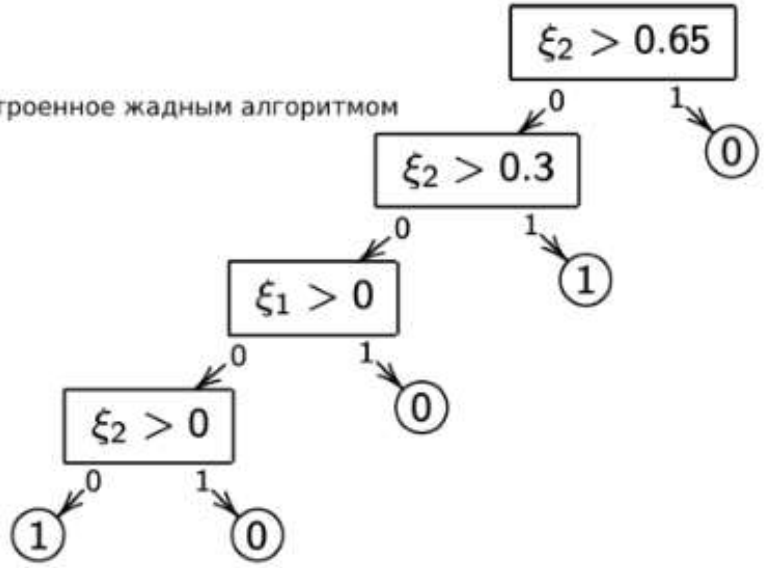


Оптимальное дерево





Дерево, построенное жадным алгоритмом



Источник: http://www.machinelearning.ru/wiki/images/archive/9/97/20140227072517!Voron-ML-Logic-slides.pdf

Критерий остановки

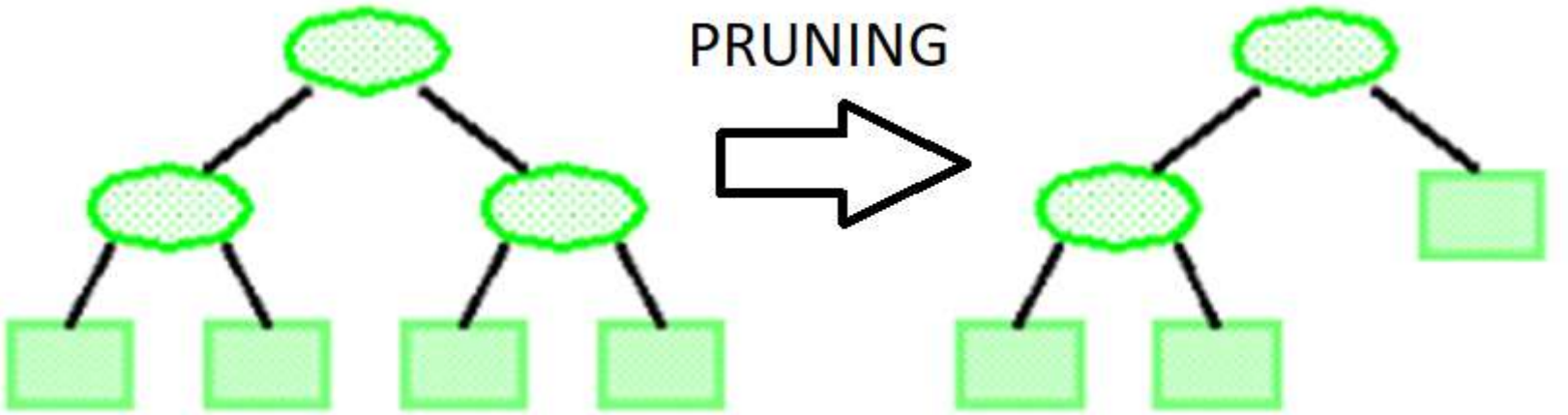
- Ограничение максимальной глубины дерева
- Ограничение минимального числа объектов в листе
- Ограничение максимального количества листьев в дереве
- Остановка в случае, если все объекты в листе относятся к одному классу
- Требование, что функционал качества при дроблении улучшался как минимум на N процентов

Можно делать на разных этапах работы алгоритма:

- *Pre-pruning* (early stopping) – проверка критериев прямо во время построения дерева
- *Pruning* – строится переобученное дерево (без ограничений), а затем выполняется стрижка (pruning), то есть удаляются некоторые вершины из дерева так, чтобы итоговое качество упало не сильно, но дерево начало подходить под условия регуляризации. Оценка качества должна выполняться на тестовой выборке.

- *Cost-complexity pruning (ccp)* – в функционал R(T) вводится штраф α за размер дерева:

$$\boldsymbol{R_\alpha(T) = R(T) + \alpha|T|}, \text{где } |\boldsymbol{T}| - \text{число листьев в дереве}$$



Работа с особенностями данных

Категориальные признаки

1. Разбивать вершину на k поддеревьев по количеству возможных значений у признака (может получиться крайне большое количество листьев)
2. Упорядочить по неубыванию доли объектов класса с $x^i = c_m$, после чего работать с ними, как со значениями вещественного признака – разделяя на значения, не превосходящие и большие определенного порога. Не нужен полный перебор. Для задачи регрессии с функцией потерь MSE значения c_m можно упорядочивать по среднему значению таргета

Обработка пропущенных значений

- При вычислении функционала делается поправка на потерю информации от исключения объектов с пропущенными признаками. Затем объекты помещаются как в левое, так и правое поддерево, им присваиваются веса, пропорциональные числу обучающих объектов в обоих поддеревьях. Прогнозы вычисляются в обоих поддеревьях и затем усредняются весами. Можно также использовать вероятность распределения классов
- Замена пропущенных значений либо на нулевое значение, либо на значения, которые превышают любое значение данного признака. Для категориальных переменных можно добавить новую категорию «пропущено»

Методы построения деревьев

- *ID3 (Iterative Dichotomiser 3)*: использует энтропийный критерий. Строит дерево до тех пор, пока в каждом листе не окажутся объекты одного класса, либо пока разбиение вершины дает уменьшение энтропийного критерия
- *C4.5* – использует критерий Gain Ratio (нормированный энтропийный критерий). Критерий остановки – ограничение на число объектов в листе. Pruning производится с помощью метода Error-based Pruning, который использует оценки обобщающей способности для принятия решения об удалении вершины. Обработка пропущенных значений осуществляется с помощью метода, который игнорирует объекты с пропущенными значениями при вычислении критерия ветвления, а затем переносит такие объекты в оба поддерева с определенными весами
- *CART (Classification And Regression Tree)* – использует критерий Джини. Pruning с помощью Cost-Complexity Pruning

Преимущества и недостатки решающих деревьев

- ✓ Интерпретируемость
- ✓ Возможность обработки пропусков в данных и категориальных признаков
- ✓ Не требуют нормализации данных
- ✓ Восстановление нелинейных зависимостей
- ✓ Эффективно работают с большими объемами данных

- × Склонность к переобучению, неустойчивость
- × Ограниченное качество классификации
- × Недостаточная эффективность при наличии трендов