

Нелинейные методы классификации

Паточенко Евгений

НИУ ВШЭ

План занятия

- Наивный байесовский классификатор
- Метод ближайших соседей (kNN)
- Отбор признаков

Наивный байесовский классификатор

Определение

Наивный байесовский классификатор (Naïve Bayes) — алгоритм обучения с учителем, основанный на применении теоремы Байеса с «наивным» предположением об условной независимости между каждой парой признаков при заданном значении переменной класса

Наивный байесовский классификатор

Теорема Байеса

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \text{ где}$$

$P(A)$ — априорная вероятность события A до наблюдения события B

$P(A|B)$ — апостериорная вероятность события A после наблюдения события B

$P(B|A)$ — вероятность наступления события B при наступлении события A

$P(B)$ — полная вероятность события B

Наивный байесовский классификатор

Описание метода

Для задачи классификации мы можем воспользоваться универсальностью теоремы и под событием A иметь в виду метку класса Y некоторого объекта, под событием B — признаки X этого объекта:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

Наивный байесовский классификатор

Описание метода

Для задачи классификации мы можем воспользоваться универсальностью теоремы и под событием A иметь в виду метку класса Y некоторого объекта, под событием B — признаки X этого объекта:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

или

$$P(y_j|x_1, x_2, \dots, x_d) = \frac{P(x_1, x_2, \dots, x_d|y_j) \cdot P(y_j)}{P(x_1, x_2, \dots, x_d)}$$

Наивный байесовский классификатор

Описание метода

Для задачи классификации мы можем воспользоваться универсальностью теоремы и под событием A иметь в виду метку класса Y некоторого объекта, под событием B — признаки X этого объекта:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

или

$$P(y_j | x_1, x_2, \dots, x_d) = \frac{P(x_1, x_2, \dots, x_d | y_j) \cdot P(y_j)}{P(x_1, x_2, \dots, x_d)}$$

Тогда в качестве ответа выдаем класс с наибольшей вероятностью:

$$y = \operatorname{argmax}_{i=1, \dots, K} \frac{P(x_1, x_2, \dots, x_d | y_j) \cdot P(y_j)}{P(x_1, x_2, \dots, x_d)}$$

Наивный байесовский классификатор

Описание метода

Так как в формуле предсказания нас интересует не точное значение вероятности, а только какой класс имеет наибольшее значение вероятности, то знаменатель можно не вычислять:

$$y = \operatorname{argmax}_{i=1,\dots,K} P(x_1, x_2, \dots, x_d | y_i) \cdot P(y_i)$$

Наивный байесовский классификатор

Описание метода

Так как в формуле предсказания нас интересует не точное значение вероятности, а только какой класс имеет наибольшее значение вероятности, то знаменатель можно не вычислять:

$$y = \operatorname{argmax}_{i=1,\dots,K} P(x_1, x_2, \dots x_d | y_j) \cdot P(y_j)$$

Считаем, что события независимы. Значит:

$$P(x_1, x_2, \dots x_d | y_j) = P(x_1 | y_j) \cdot P(x_2 | y_j) \cdot \dots \cdot P(x_d | y_j)$$

Наивный байесовский классификатор

Гауссов наивный байесовский классификатор

Часто используется гауссов наивный байес
(приближает численные признаки нормальным распределением):

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Параметры μ_y и σ_y оцениваются с использованием метода максимального правдоподобия

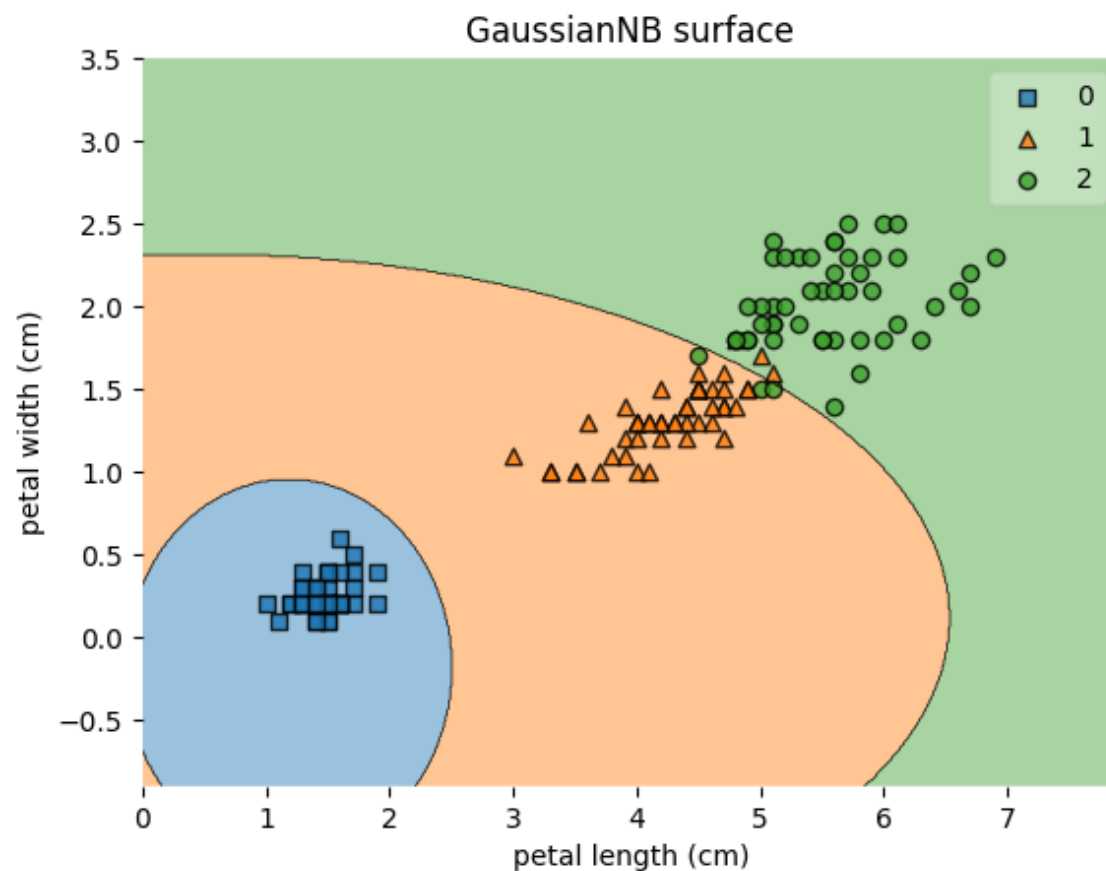
Наивный байесовский классификатор

Примеры других реализаций

- многочленный наивный байес — для полиномиально распределенных данных и классификации текста (частая задача — определение спама)
- наивный байесовский алгоритм дополнения — для несбалансированных данных
- наивный байес Бернулли — для данных с многомерным распределением Бернулли
- категориальный наивный байес — для категориальных данных

[Документация scikit-learn](#)

Наивный байесовский классификатор



Пример результата работы алгоритма по классификации сортов ириса

Наивный байесовский классификатор

Применимость

Преимущества:

Наивный байесовский классификатор

Применимость

Преимущества:

- Простота реализации

Наивный байесовский классификатор

Применимость

Преимущества:

- Простота реализации
- Низкие вычислительные затраты

Наивный байесовский классификатор

Применимость

Преимущества:

- Простота реализации
- Низкие вычислительные затраты
- Если признаки действительно независимы, метод оптимален

Наивный байесовский классификатор

Применимость

Преимущества:

- Простота реализации
- Низкие вычислительные затраты
- Если признаки действительно независимы, метод оптимален

Недостатки:

Наивный байесовский классификатор

Применимость

Преимущества:

- Простота реализации
- Низкие вычислительные затраты
- Если признаки действительно независимы, метод оптимален

Недостатки:

- В большинстве реальных задач признаки не полностью независимы, поэтому алгоритм показывает низкое качество

Метод ближайших соседей (kNN)

Определение

Метод ближайших соседей (k-nearest neighbors, kNN) — метрический алгоритм классификации или регрессии, основанный на *гипотезе компактности*: похожие объекты лежат в одном классе (или имеют схожее значение целевой переменной в задачах регрессии)

Алгоритм не строит модель на этапе обучения, а просто запоминает всю обучающую выборку (lazy learning). Все вычисления происходят только на этапе прогнозирования

Метод ближайших соседей (kNN)

Логика работы

Хотим предсказать класс



Метод ближайших соседей (kNN)

Логика работы

Чтобы классифицировать новый объект, нужно:

- вычислить расстояние до каждого из объектов обучающей выборки
- выбрать k объектов обучающей выборки, расстояние до которых минимально

Класс нового объекта – это класс, наиболее часто встречающийся среди k ближайших соседей

Метод ближайших соседей (kNN)

Логика работы

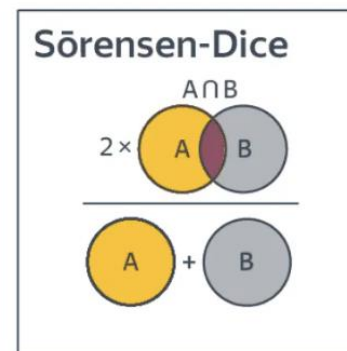
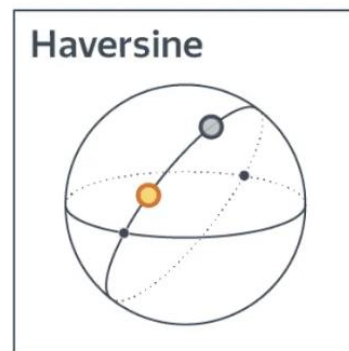
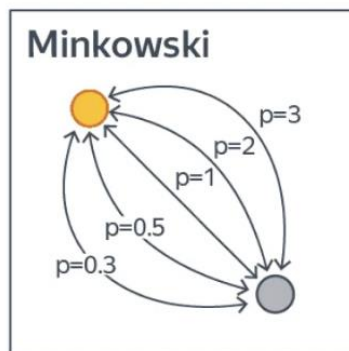
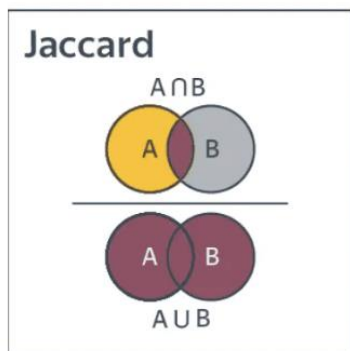
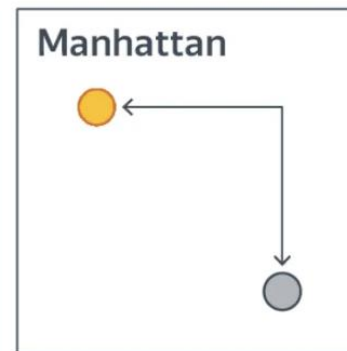
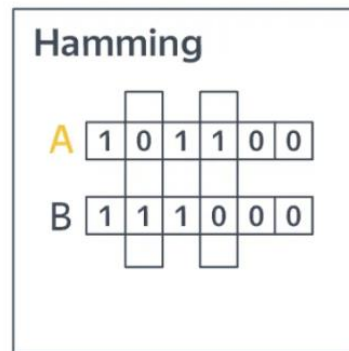
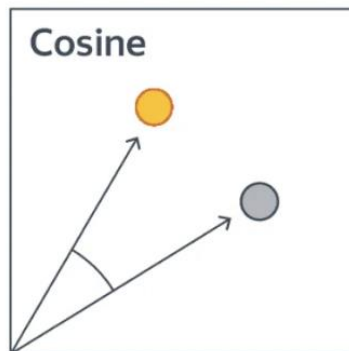
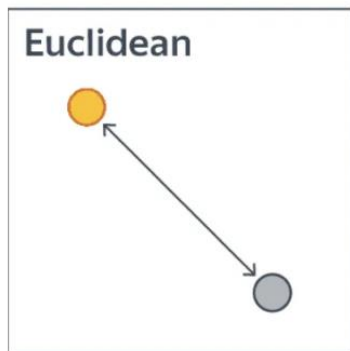
Чтобы классифицировать новый объект, нужно:

- **вычислить расстояние до каждого из объектов обучающей выборки**
- выбрать k объектов обучающей выборки, расстояние до которых минимально

Класс нового объекта – это класс, наиболее часто встречающийся среди k ближайших соседей

Метод ближайших соседей (kNN)

Метрики



Метод ближайших соседей (kNN)

Метрики

- Наиболее частый выбор, если значения признаков непрерывные, – **евклидово**

расстояние: $\sqrt{\sum_{i=1}^d (a_i - b_i)^2}$

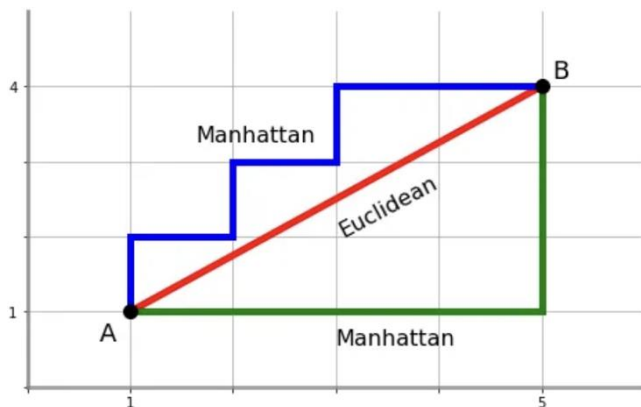
Метод ближайших соседей (kNN)

Метрики

- Наиболее частый выбор, если значения признаков непрерывные, – **евклидово**

расстояние: $\sqrt{\sum_{i=1}^d (a_i - b_i)^2}$

- Лучше работает на данных с выбросами — **манхэттенская метрика:** $\sum_i |a_i - b_i|$



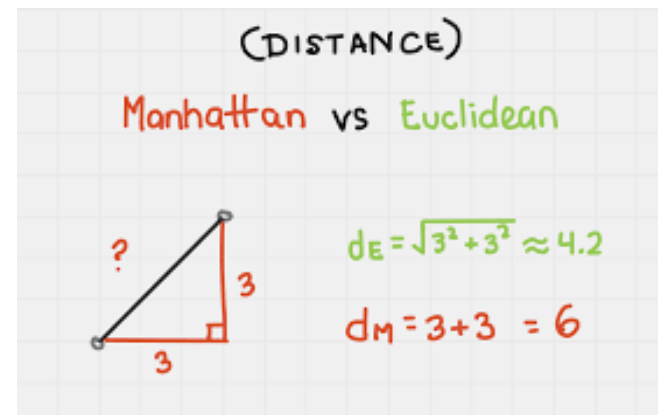
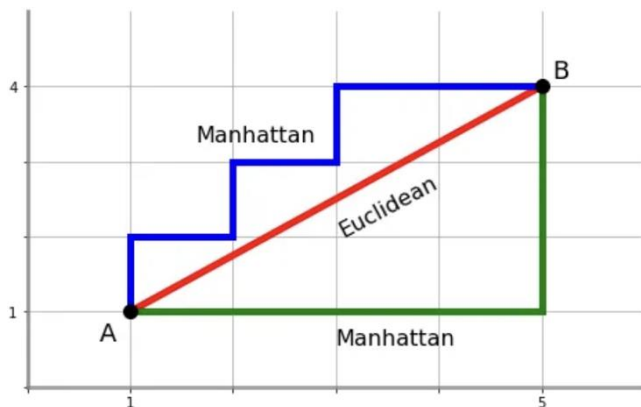
Метод ближайших соседей (kNN)

Метрики

- Наиболее частый выбор, если значения признаков непрерывные, – **евклидово**

расстояние: $\sqrt{\sum_{i=1}^d (a_i - b_i)^2}$

- Лучше работает на данных с выбросами — **манхэттенская метрика:** $\sum_i |a_i - b_i|$



Метод ближайших соседей (kNN)

Метрики

- Обобщение евклидова расстояния и манхэттенского — **метрика Минковского**:

$$\sqrt[p]{\sum_i |a_i - b_i|^p}$$

Метод ближайших соседей (kNN)

Метрики

- Обобщение евклидова расстояния и манхэттенского — **метрика Минковского**:

$$\sqrt[p]{\sum_i |a_i - b_i|^p}$$

- Частный случай метрики Минковского для категориальных признаков — **расстояние**

Хэмминга: $\sum_{k=1}^p |a_{ik} - b_{jk}|$

Метод ближайших соседей (kNN)

Метрики

- Обобщение евклидова расстояния и манхэттенского — **метрика Минковского**:

$$\sqrt[p]{\sum_i |a_i - b_i|^p}$$

- Частный случай метрики Минковского для категориальных признаков — **расстояние**

Хэмминга: $\sum_{k=1}^p |a_{ik} - b_{jk}|$

- Для текстовых данных — **косинусное расстояние**: $1 - \cos \theta = 1 - \frac{a \cdot b}{\|a\| \|b\|}$

Метод ближайших соседей (kNN)

Логика работы

Чтобы классифицировать новый объект, нужно:

- вычислить расстояние до каждого из объектов обучающей выборки
- **выбрать k объектов обучающей выборки, расстояние до которых минимально**

Класс нового объекта – это класс, наиболее часто встречающийся среди k ближайших соседей

Метод ближайших соседей (kNN)

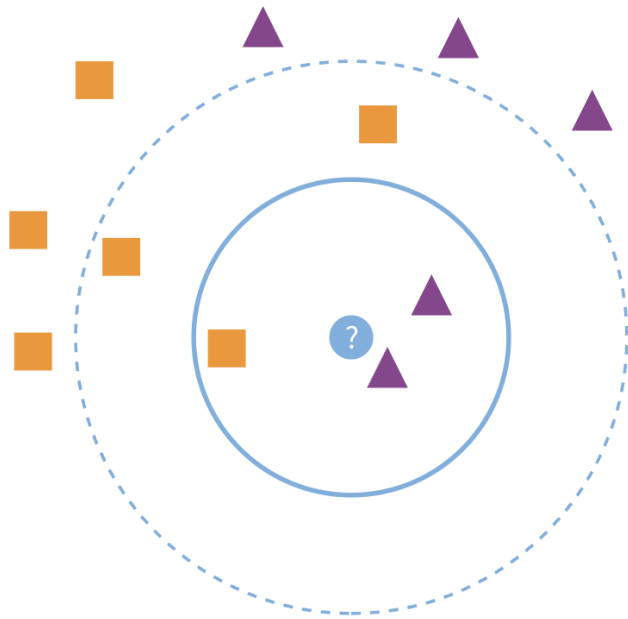
Гиперпараметр k

Мы сами выбираем значение гиперпараметра k — числа ближайших соседей, на которые мы смотрим для определения класса

Метод ближайших соседей (kNN)

Гиперпараметр k

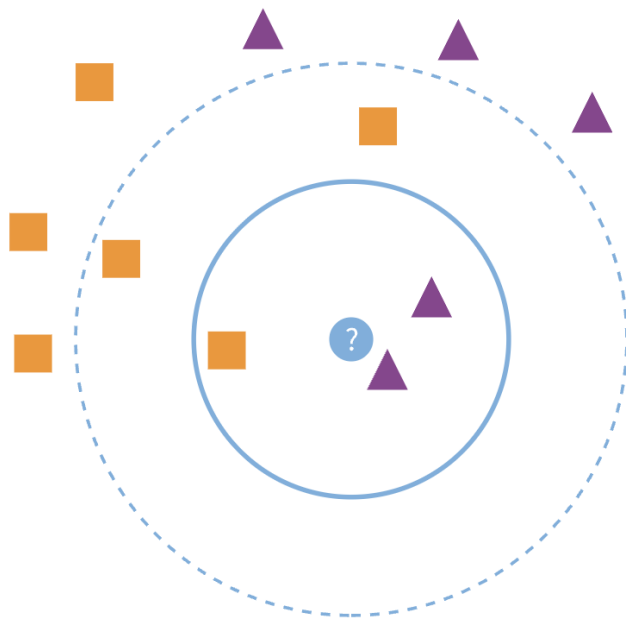
В зависимости от выбранного значения k результаты классификации будут меняться



Метод ближайших соседей (kNN)

Гиперпараметр k

В зависимости от выбранного значения k результаты классификации будут меняться

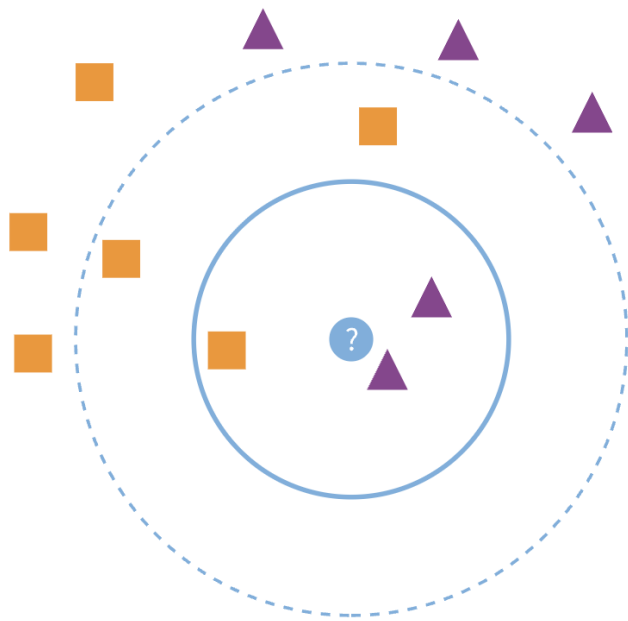


К какому классу будет отнесен объект при $k = 3$?

Метод ближайших соседей (kNN)

Гиперпараметр k

В зависимости от выбранного значения k результаты классификации будут меняться



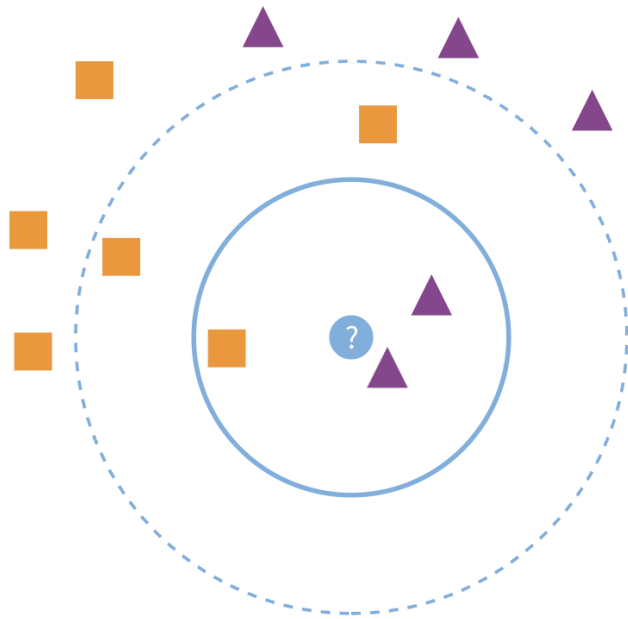
К какому классу будет отнесен объект при $k = 3$?

Треугольник

Метод ближайших соседей (kNN)

Гиперпараметр k

В зависимости от выбранного значения k результаты классификации будут меняться



К какому классу будет отнесен объект при $k = 3$?

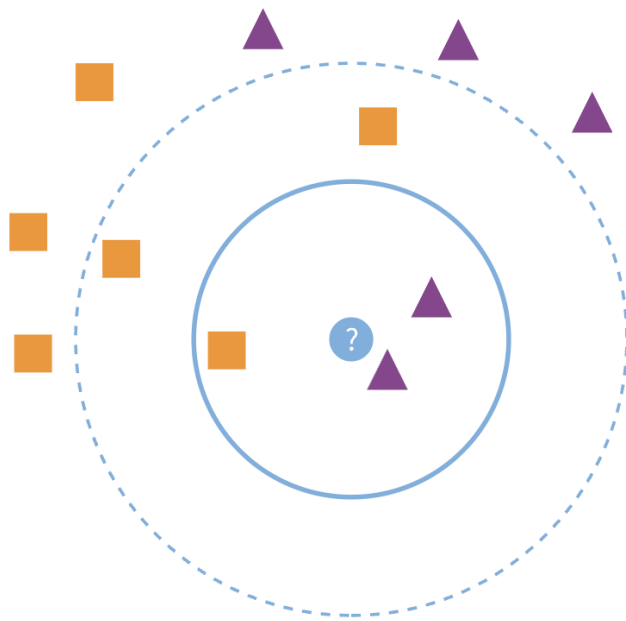
Треугольник

К какому классу будет отнесен объект при $k = 5$?

Метод ближайших соседей (kNN)

Гиперпараметр k

В зависимости от выбранного значения k результаты классификации будут меняться



К какому классу будет отнесен объект при $k = 3$?

Треугольник

К какому классу будет отнесен объект при $k = 5$?

Квадрат

Метод ближайших соседей (kNN)

Гиперпараметр k

Как будет работать алгоритм при $k = 1$?

Метод ближайших соседей (kNN)

Гиперпараметр k

Как будет работать алгоритм при $k = 1$?

Слишком малое k создает эффект переобучения. При $k = 1$ алгоритм будет присваивать любому новому наблюдению метку класса ближайшего объекта

Метод ближайших соседей (kNN)

Гиперпараметр k

Как будет работать алгоритм при $k = 1$?

Слишком малое k создает эффект переобучения. При $k = 1$ алгоритм будет присваивать любому новому наблюдению метку класса ближайшего объекта

При больших значениях k уменьшается влияние шумов в данных, но снижается выраженность границ классов и классификация становится слишком грубой

Метод ближайших соседей (kNN)

Гиперпараметр k

Как будет работать алгоритм при $k = 1$?

Слишком малое k создает эффект переобучения. При $k = 1$ алгоритм будет присваивать любому новому наблюдению метку класса ближайшего объекта

При больших значениях k уменьшается влияние шумов в данных, но снижается выраженность границ классов и классификация становится слишком грубой

Выбор значения k сводится к компромиссу между точностью и обобщающей способностью модели

Метод ближайших соседей (kNN)

Формализация

Пусть k – количество соседей. Для каждого объекта u возьмем k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}$$

Тогда класс объекта u определяется следующим образом:

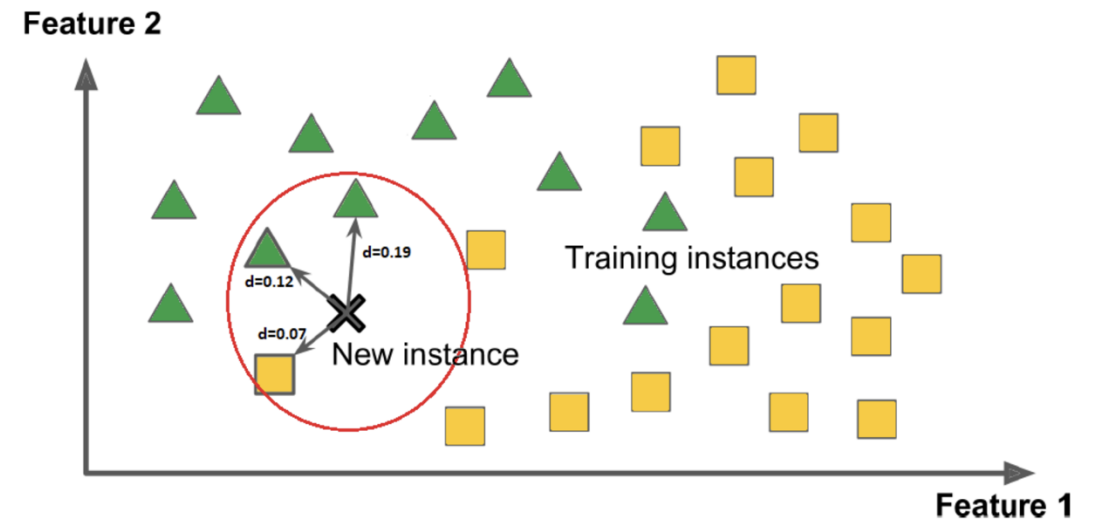
$$\operatorname{argmax}_{y \in Y} \sum_{i=1}^k I[y(x_{(i;u)}) = y]$$

Метод ближайших соседей (kNN)

Взвешенный kNN

Чем ближе объекты в пространстве признаков, тем выше их сходство. Для учета близости при предсказании класса нового объекта используем веса: ближайшие соседи получают больший вес:

$$\operatorname{argmax}_{y \in Y} \sum_{i=1}^k \frac{1}{p(u, x_{(i;u)})} I[y(x_{(i;u)}) = y]$$



Метод ближайших соседей (kNN)

Применимость

Преимущества:

- Простота реализации
- Не требует обучения (lazy learning)
- Высокая интерпретируемость

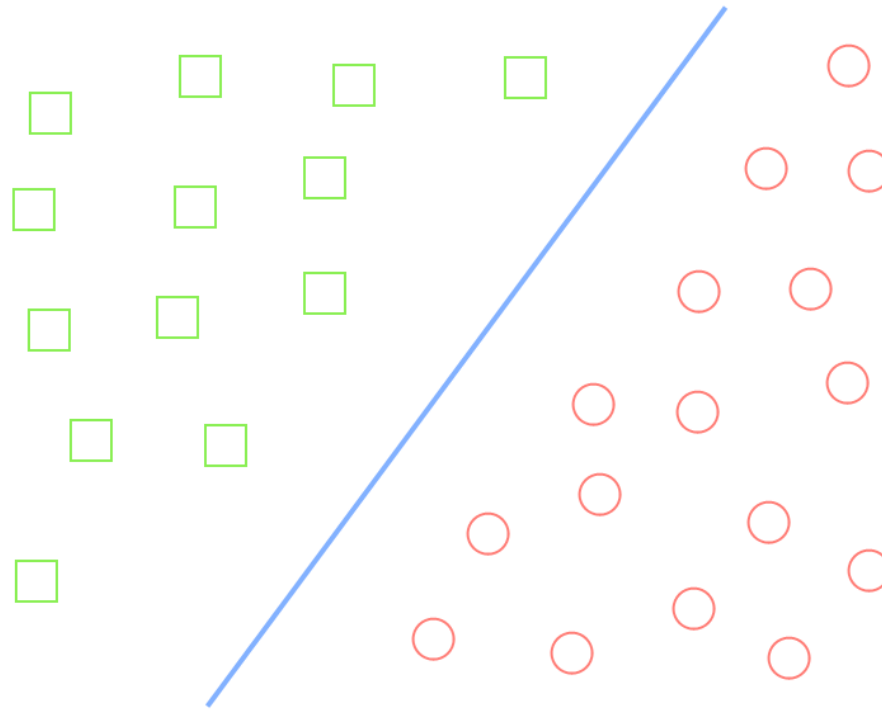
Недостатки:

- Дорогой по вычислениям на этапе предсказания (нужно считать расстояния до всех объектов)
- Чувствителен к масштабу признаков (требуется нормализация)

SVM (напоминание)

Линейно разделимая выборка

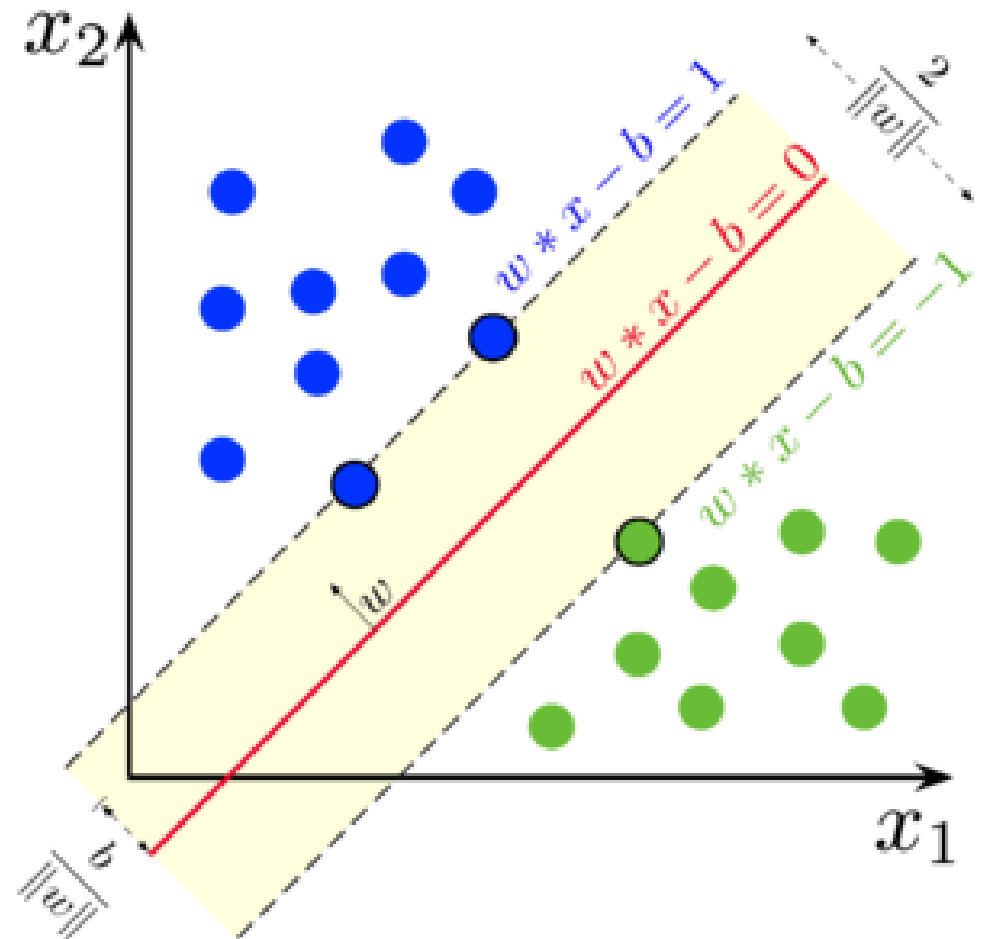
Выборка линейно разделима, если существует такой вектор параметров β , что соответствующий классификатор $f(x)$ не допускает ошибок на этой выборке



SVM (напоминание)

*SVM (Support Vector Machine) для
линейно разделимого случая*

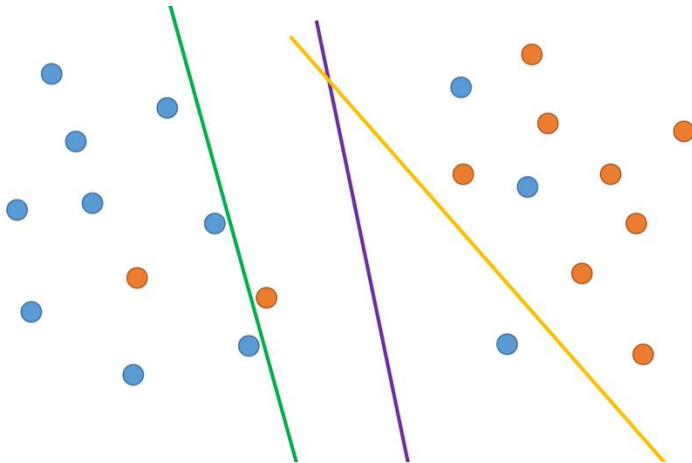
Цель SVM — максимизировать ширину
разделяющей полосы, то есть найти
наилучшую разделяющую
гиперплоскость, которая максимально
отдалена от ближайших точек каждого
класса (**опорных векторов**)



Kernel SVM

Линейно неразделимая выборка

В реальных задачах в исходном пространстве признаков данные гораздо чаще *линейно неразделимы*

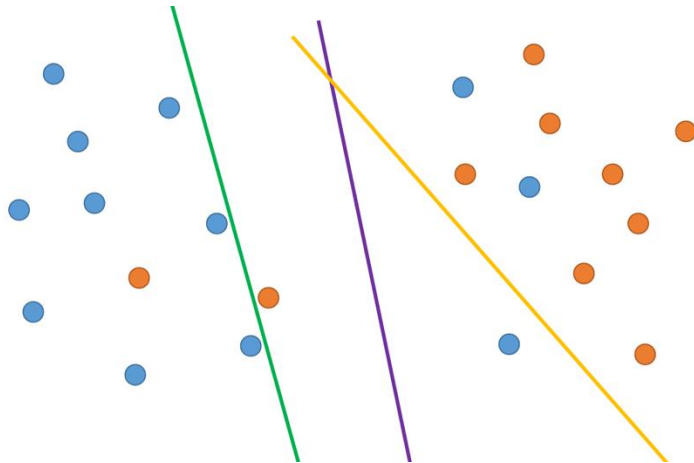
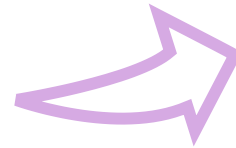


Kernel SVM

Линейно неразделимая выборка

В реальных задачах в исходном пространстве признаков данные гораздо чаще *линейно неразделимы*

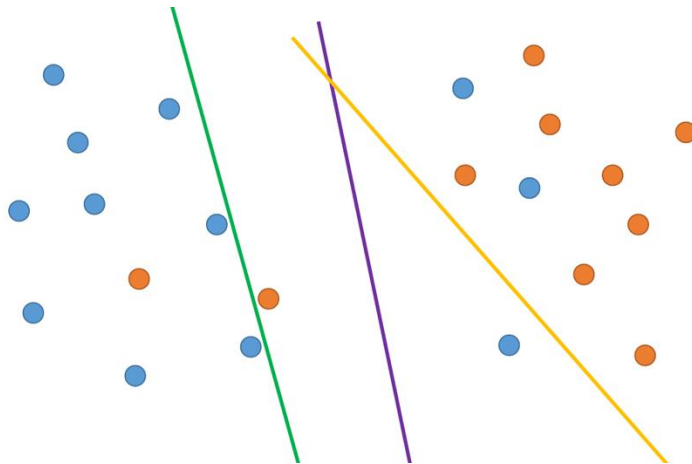
Возможное решение — добавить полиномиальные признаки



Kernel SVM

Линейно неразделимая выборка

В реальных задачах в исходном пространстве признаков данные гораздо чаще *линейно неразделимы*



Возможное решение — добавить полиномиальные признаки

Проблема: количество признаков в результате такого преобразования может увеличиться в сотни раз, что кратно повышает вычислительные расходы

Kernel SVM

Kernel trick

Прием, позволяющий использовать SVM в нелинейных задачах, при котором скалярное произведение трансформированных векторов n -й степени заменяется на их произведение в степени n :

Вместо явного добавления полиномиальных или других функций признаков используется *ядро* $K(a, b)$

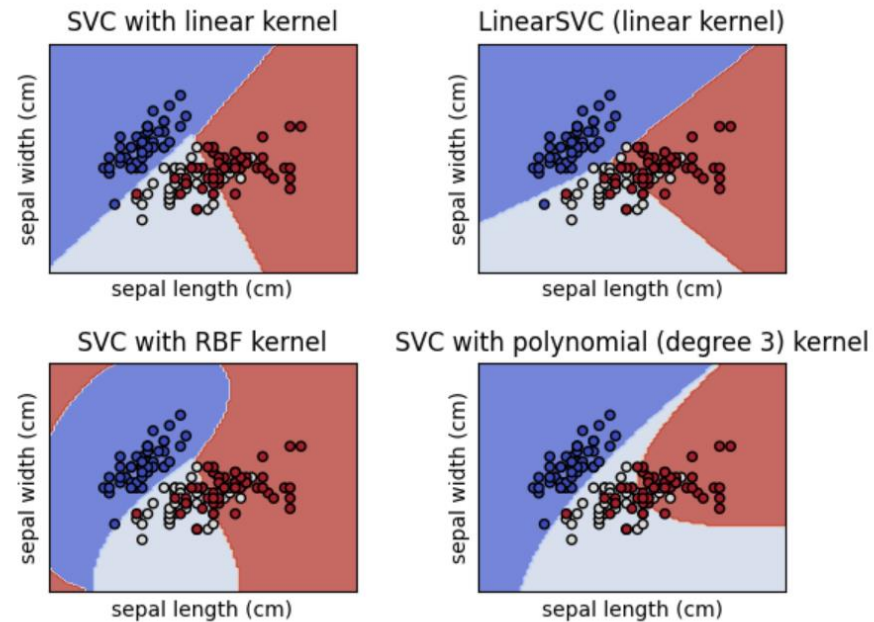
Ядро вычисляет скалярное произведение преобразованных векторов $\phi(a), \phi(b)$ напрямую: $\phi(a)^T \cdot \phi(b) = (a^T \cdot b)^n$, где ϕ — полиномиальная функция степени n

Это позволяет строить сложные границы без взрывного роста размерности и вычислительной сложности

Kernel SVM

Ядро

Ядро — это функция, определяющая сходство между объектами в исходном пространстве



Kernel SVM

Примеры ядер

- **Линейное:** $K(a, b) = a^T \cdot b$

Обычный линейный классификатор

Kernel SVM

Примеры ядер

- **Линейное:** $K(a, b) = a^T \cdot b$

Обычный линейный классификатор

- **Полиномиальное:** $K(a, b) = (\gamma a^T \cdot b + r)^d$

Улавливает более сложные зависимости

Степень d управляет сложностью границы

γ — параметр, определяющий радиус влияния обучающих образцов. Чем меньше γ , тем более гладкая граница

r — коэффициент сдвига, который влияет на гибкость ядра и сложность решающей границы

Kernel SVM

Примеры ядер

- **Гауссовское RBF (радиально-базисная функция):** $K(a, b) = \exp(-\gamma \|a - b\|^2)$

Менее подвержено переобучению и хорошо подходит для данных сложной нелинейной формы, учитывает не только значения признаков, но и их распределение

Kernel SVM

Примеры ядер

- **Гауссовское RBF (радиально-базисная функция):** $K(a, b) = \exp(-\gamma \|a - b\|^2)$

Менее подвержено переобучению и хорошо подходит для данных сложной нелинейной формы, учитывает не только значения признаков, но и их распределение

- **Сигмоидальное:** $K(a, b) = \tanh(\gamma a^T \cdot b + r)$

Имитирует использование двухслойной нейронной сети с сигмоидальной функцией активации. Хорошо работает со сложными нелинейными случаями, но может переобучаться в случае появления шумов и выбросов

Спасибо за внимание!

