Лекция 2

Непараметрические методы

Машинное обучение

Андрей Фильченков / Сергей Муравьев

11.09.2020

План лекции

- Валидация моделей
- Классификация и регрессия на основе похожести
- Метод одного ближайшего соседа
- Метод k ближайших соседей (kNN)
- Обобщенный метрический классификатор
- Непараметрическая регрессия
- Оценка качества

- В презентации используются материалы курса «Машинное обучение» К.В. Воронцова
- Слайды доступны: **shorturl.at/ltVZ3**
- Видео доступны: **shorturl.at/hjyAX**

План лекции

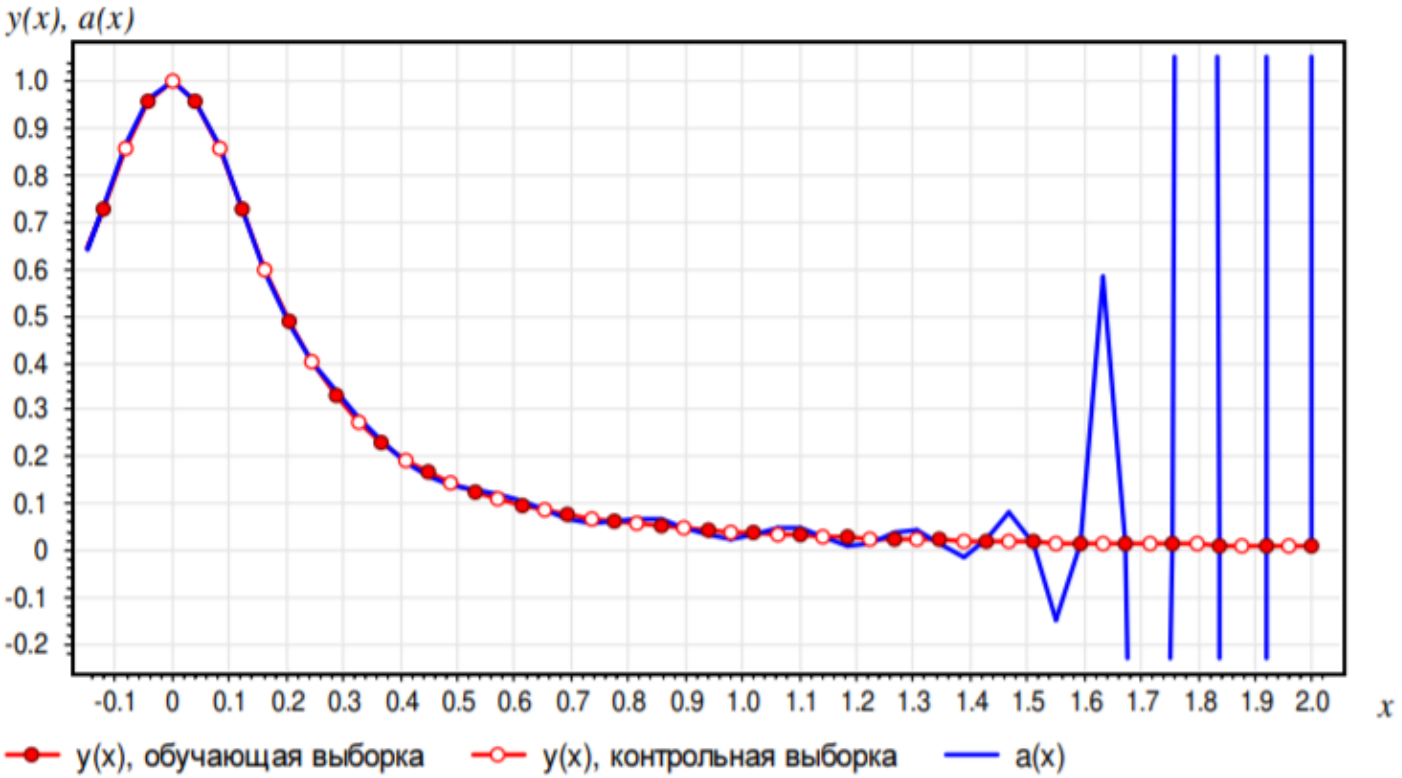
- **Валидация моделей**
- Классификация и регрессия на основе похожести
- Метод одного ближайшего соседа
- Метод *k* ближайших соседей (kNN)
- Обобщенный метрический классификатор
- Непараметрическая регрессия
- Оценка качества

Проблема переобучения (напоминание)

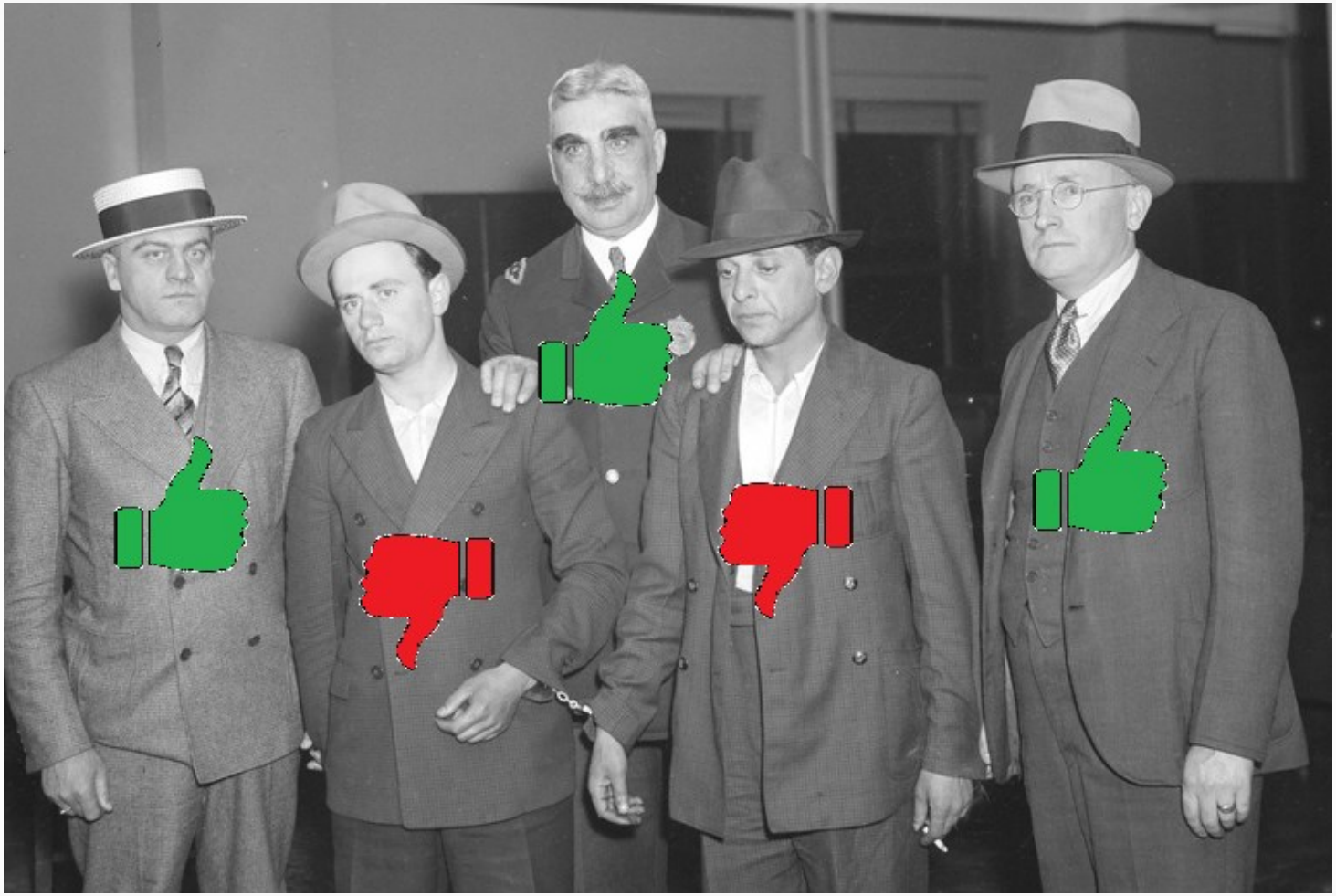
Проблема переобучения — начиная с определенного уровня сложности предсказательной модели, чем лучше алгоритм показывает себя на тренировочном наборе данных $\mathcal{D}$, тем хуже он работает на реальных объектах.

Переобученный алгоритм (напоминание)

$y(x) = \dfrac{1}{1 + 25x^2}$; $a(x)$ — многочлен степени $n = 38$



Ложные зависимости



Декомпозиция метода обучения

Метод обучения — это отображение

$$\mu = \mu^{val} \cdot \mu^{A} : (X \times Y)^{\dim} \to A,$$

которое возвращает алгоритм $a \in A$ для набора данных $\mathcal{D} \in (X \times Y)^{\dim}$,

где $(X \times Y)^{\dim} = \bigcup_{i \in \dim N \subseteq \mathbb{N}} (X \times Y)^{i}$

μ^{val} выбирается и применяется независимо от μ^{A} (хотя они тесно связаны).

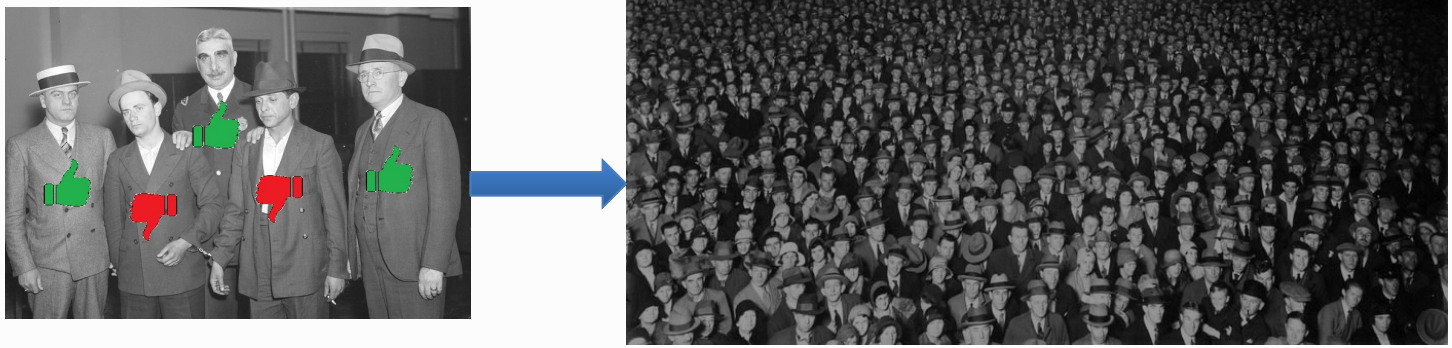
Способы борьбы с переобучением

- Аналитический
  - Регуляризация
  - Выбор параметрического семейства алгоритмов
  - Дизайн настройки алгоритмов
- Эмпирический
  - Внешние меры валидации
  - Внутренние меры валидации

Внешние меры валидации

Идея в оценке обобщающей способности μ^A на имеющихся данных

вместо



будем

Валидация на отложенной выборке

Hold-out validation, HO

Разобьем обучающее множество на две части:

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$$

Обучение (train), $\mathcal{D}_{train}$	Тестирование (test), $\mathcal{D}_{test}$

Будем решать следующую задачу:

$$\mu_{\text{HO}}^{val}(\mu_A, \mathcal{D}_{train}, \mathcal{D}_{test}) = \text{argmin}_{\mu^A} \mathcal{L}(\mu^A(\mathcal{D}_{train}), \mathcal{D}_{test})$$

Полная кросс-валидация

Зафиксируем значения r и e:

$$|\mathcal{D}_{train}| = r, |\mathcal{D}_{test}| = e.$$

Разобьем $\mathcal{D}$ всеми возможными способами на $\mathcal{D}_{train}$ и $\mathcal{D}_{test}$ соответствующего размера

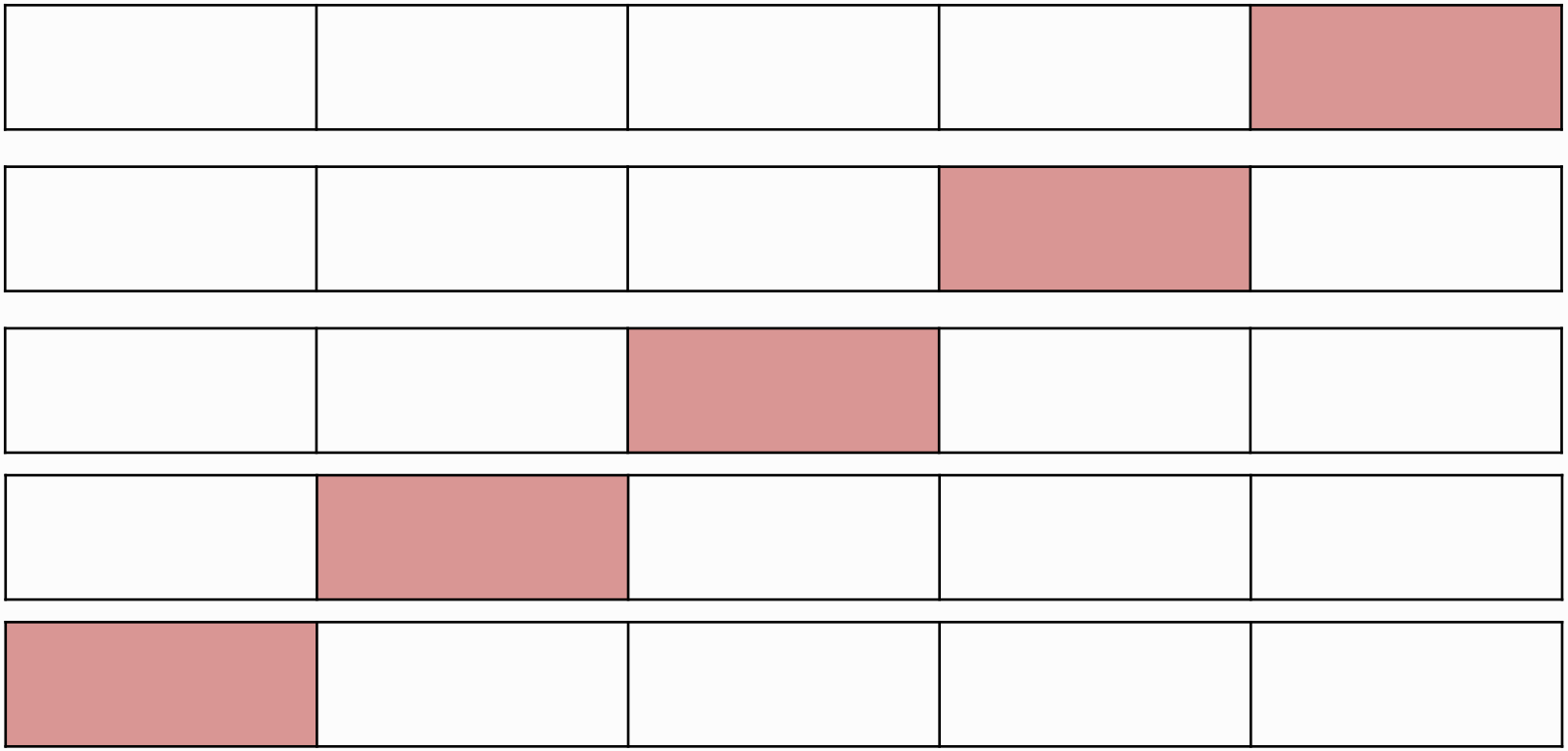
Будем решать следующую задачу:

$$\mu_{\text{CCV}}^{val}(\mathcal{D}, r, e) = \frac{1}{C_{e+r}^{e}} \text{argmin}_{\mu^A} \sum_{\mathcal{D}=\mathcal{D}_{train} \cup \mathcal{D}_{test}} \mathcal{L}(\mu^A(\mathcal{D}_{train}), \mathcal{D}_{test})$$

В чем проблема такого метода?

Кросс-валидация

Cross-validation (скользящий контроль)
Разобьем обучающий набор данных k раз на k частей



Кросс-валидация по k блокам

k-fold cross-validation

Каждый из k блоков ровно один раз оказывается тестирующим.

k обычно 10 (бывает 5, в крайнем случае 3).

Разобьем $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_k$, $|\mathcal{D}_i| \approx |\mathcal{D}_j|$.

Будем решать следующую задачу:

$$\mu_{\text{CV}}^{val}(\mathcal{D}, k) = \frac{1}{k} \text{argmin}_{\mu^A} \sum_{i=1}^{k} \mathcal{L}(\mu^A(\mathcal{D} \backslash \mathcal{D}_i), \mathcal{D}_i)$$

t кросс-валидаций по k блокам

$t \times k$-fold cross-validation

Повторим t раз:
разобьем выборку на k блоков, каждый из k блоков ровно один раз оказывается в тестовом множестве.

t обычно равно 10 (но все зависит от времени, которое есть на тестирование).

Кросс-валидация по отдельным объектам

Leave-one-out cross-validation, LOO

Разобьем обучающее множество на $|\mathcal{D}| - 1$ и 1 объекты $|\mathcal{D}|$ раз.

Будем решать следующую задачу:

$$\mu_{\text{LOO}}^{val}(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\text{argmin}_{\mu^A}\sum_{i=1}^{|\mathcal{D}|}\mathcal{L}(\mu^A(\mathcal{D}\backslash p_i), p_i)\,.$$

где $p_i = (x_i, y_i)$.

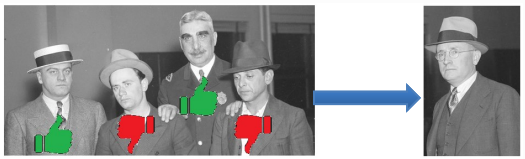
Стратифицированная кросс-валидация по k блокам

Stratified k-fold cross-validation

В случае, если объекты каких-то классов или с какими-то значениями признаков недостаточно часто встречаются в выборке, то лучше разбивать на блоки так, чтобы в каждом блоке эта статистика была пропорциональна статистике в выборке.

Очень сложный вопрос

Кросс-валидация по пяти блокам дает пять моделей

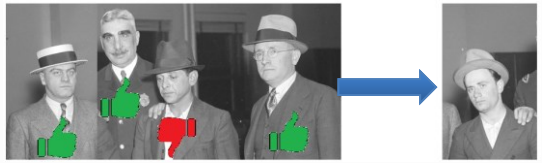


a_{θ_1}

a_{θ_2}

a_{θ_3}

a_{θ_4}



a_{θ_5}

Какую из них в итоге использовать?

Очень сложный вопрос

a_{θ_1} a_{θ_2} a_{θ_3} a_{θ_4} a_{θ_5}

Какую из них в итоге использовать?
Никакую! Надо обучить новую модель на всем наборе данных.

Еще один очень сложный вопрос

a_{θ_1} a_{θ_2} a_{θ_3} a_{θ_4} a_{θ_5}

Какую из них в итоге использовать?
Никакую! Надо обучить новую модель на всем наборе данных.

А почему мы так не переобучимся?

Что оценивает валидация

- Мы оцениваем не модели, а μ^A
- Для любого набора данных (в разумных пределах) мы должны быть способны построить хорошо обобщающие предсказательные модели.
- Тот метод, который строит лучшие модели в этом смысле, и есть лучший. Именно его нужно применять как для всех имеющихся данных, так и для новых данных.

Что еще приводит к переобучению

- Смещенный (нерепрезентативный) набор данных
- Плохо подобранная метрика качества измерения алгоритмов (например, точность для редких классов)
- Систематические смещения в методах валидации и обучения
- Непонимание скрытых гиперпараметров

Кросс-валидация в этих случаях бессильна

План лекции

- Валидация моделей
- **Классификация и регрессия на основе похожести**
- Метод одного ближайшего соседа
- Метод *k* ближайших соседей (kNN)
- Обобщенный метрический классификатор
- Непараметрическая регрессия
- Оценка качества

Формулировка задачи

X — множество объектов, Y — множество меток,

$y : X \to Y$ — неизвестная зависимость, $|Y| \ll \infty$

$\mathcal{D} = \{(x_i, y_i)\}$ — обучающее множество.

Задача: найти алгоритм $a : X \to Y$, приближающий y на X.

Что это за задача с точки зрения машинного обучения?

Задача классификации (напоминание)

X — множество объектов, Y — множество меток,

$y : X \to Y$ — неизвестная зависимость, $|Y| \ll \infty$

$\mathcal{D} = \{(x_i, y_i)\}$ — обучающее множество.

Задача: найти алгоритм $a : X \to Y$, приближающий y на X.

Что это за задача с точки зрения машинного обучения?

Классификация, потому что $|Y| \ll \infty$.

Утиный тест

Утиный тест (duck test):
Если нечто выглядит как утка, плавает как утка и крякает как утка, то это, вероятно, и есть утка.

Тестирование утиного теста

Если нечто выглядит как утка, плавает как утка и крякает как утка, то это, вероятно, и есть утка.

Выглядит	Плавает	Крякает	Это утка?
Как утка	Как утка	Как утка	Вероятно, да
Вообще не как утка	Отдаленно похоже на утку	Не как утка	Вероятно, нет

Утиный тест как алгоритм классификации

Как выглядели обучающие данные:

Много уток, много не уток (неуток).

Как строился классификатор:

1. Для уток выделили **ключевые признаки**.
2. **Понятие похожести** используется для оценки близости признаков.
3. Логический сепаратор используется для классификации.

Основная идея

Ключевая гипотеза: похожие объекты принадлежат одному классу / имеют похожие значения целевой функции.

Основная идея для классификации: будем относить новый объект к тому классу, на объекты которого он похож.

- Рассуждение по аналогии
- Ленивое обучение

План лекции

- Валидация моделей
- Классификация и регрессия на основе похожести
- **Метод (одного) ближайшего соседа**
- Метод *k* ближайших соседей (kNN)
- Обобщенный метрический классификатор
- Непараметрическая регрессия
- Оценка качества

Формализация «похожести»

«Похожесть» — это мера сходства между объектами. Мы будем говорить про **расстояния** (метрики).

Метрическое пространство это множество с заданной на нем метрикой

$$\rho(x, y): X \times X \to [0; +\infty)$$

симметрия, неравенство треугольна, различимость нетождественных объектов, неразличимость тождественных объектов.

Часто используемые метрики

Расстояние Минковского:

$$p(x, y) = \left(\sum_i |x_i - y_i|^p\right)^{\frac{1}{p}},$$

при $p = 2$, это Евклидово расстояние;
при $p = 1$, это Манхеттенское расстояние.

Косинусное расстояние:

$$p(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Расстояние Махаланобиса:

$$p(x, y) = \sqrt{(x - y)^\top S^{-1}(x - y)},$$

где S — матрица ковариации между x и y.

Метод (одного) ближайшего соседа (1NN)

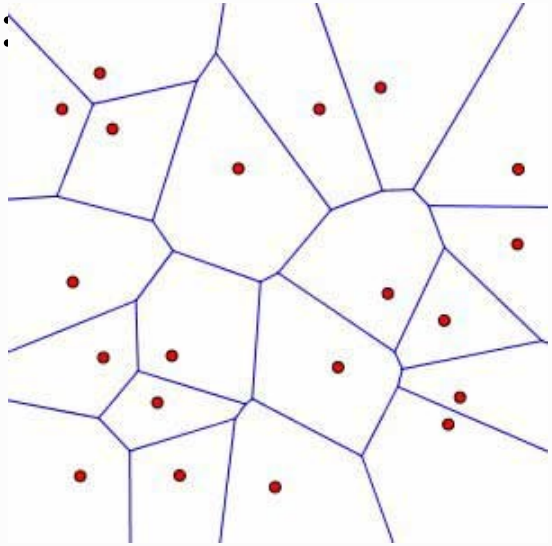
$x_{(u,1)}$ — **ближайший сосед** объекта u:

$$x_{(u,1)} = \operatorname{argmin}_{x \in X_{train}} \rho(u, x).$$

Классификатор **(одного) ближайшего соседа**:

$$a_{1\mathrm{NN}}(u, \mathcal{D}_{train}) = y\left(x_{(u,1)}\right).$$

Диаграмма Вороного:



Анализ 1NN

Достоинства:

- простота реализации;
- понятность;
- интерпретируемость.

Недостатки:

- чувствительность к шуму;
- низкое качество работы;
- нет обучаемых параметров (явно заданных);
- необходимость хранить все объекты.

1NN для регрессии

Как бы выглядел 1NN для регрессии?

1NN для регрессии

Как бы выглядел 1NN для регрессии?

Вернуть значение ближайшего соседа

Это хоть сколько-нибудь полезно?

1NN для регрессии

Как бы выглядел 1NN для регрессии?

Вернуть значение ближайшего соседа

Это хоть сколько-нибудь полезно?

На самом деле, работает не столь уж плохо, но никто не использует

План лекции

- Валидация моделей
- Классификация и регрессия на основе похожести
- Метод одного ближайшего соседа
- **Метод *k* ближайших соседей (kNN)**
- Обобщенный метрический классификатор
- Непараметрическая регрессия
- Оценка качества

Метод k ближайших соседей (kNN)

Выберем расстояние ρ.

Отсортируем объекты:

$$\rho\big(u, x_{(u,1)}\big) \leq \rho\big(u, x_{(u,2)}\big) \leq \cdots \leq \rho\big(u, x_{(u,|\mathcal{D}_{train}|)}\big).$$

Алгоритм kNN:

$$a_{k\text{NN}}(u; \mathcal{D}_{train}) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^{k} \big[y(x_{(u,i)}) = y\big]$$

$$a_{k\text{NN}}(u; \mathcal{D}_{train}) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^{|\mathcal{D}_{train}|} \big[y(x_{(u,i)}) = y\big][i \leq k]$$

План лекции

- Валидация моделей
- Классификация и регрессия на основе похожести
- Метод одного ближайшего соседа
- Метод *k* ближайших соседей (kNN)
- **Обобщенный метрический классификатор**
- Непараметрическая регрессия
- Оценка качества

Как можно улучшить 1NN?

- Более сложная модель (больше параметров)
- Выбор расстояния
- Уменьшение размерности
- Использование эффективных структур для хранения данных
- Прореживание объектов
- Фильтрация шума
- Выбор прототипов

Обобщенный метрический классификатор

$$a_{\text{GenDistClassifier}}(u; \mathcal{D}_{train}) =$$

$$= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{|\mathcal{D}_{train}|} [y(x_{(u,i)}) = y] w_{(i,u)},$$

где $w_{(i,u)}$ функция значимости i-го соседа u.

Можно думать про $\sum_{i=1}^{|\mathcal{D}_{train}|} [y(x_{(u,i)}) = y] w_{(i,u)}$ как оценку близости объекта u к классу y.

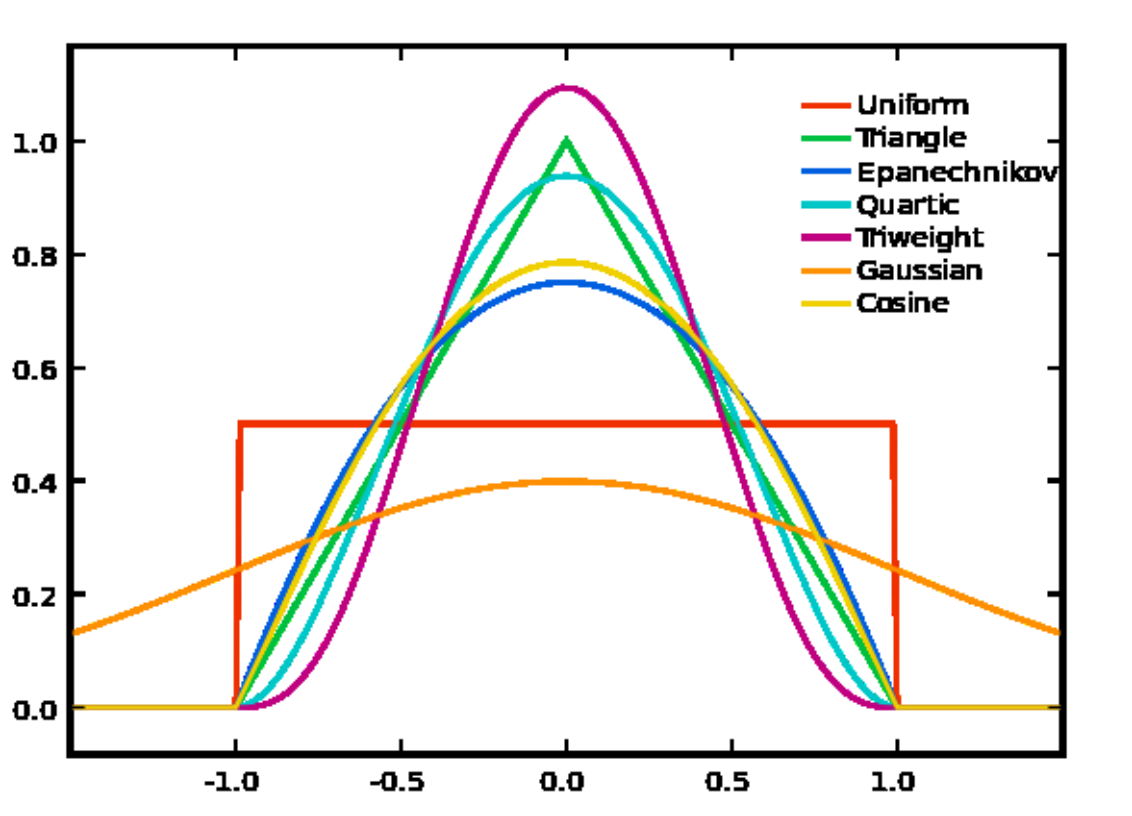
Как выбирать w?

$w_{(i,u)}$:

- линейно убывающая функция;
- экспоненциально убывающая функция;
- ядерная функция.

Ядерная функция

Ядерная функция $K(x)$ это симметричная неотрицательная функция, $\int_{-\infty}^{+\infty} K(x)\,dx = 1$.



Окно Парзена-Розенблатта

С фиксированной шириной окна:

$$a_{\text{GenDistClass}h}(u; \mathcal{D}_{train}; h; K) =$$

$$= \text{argmax}_{y \in Y} \sum_{i=1}^{|\mathcal{D}_{train}|} [y(x_{(u,i)}) = y] K\left(\frac{\rho(u, x_{(u,i)})}{h}\right),$$

С нефиксированной шириной окна:

$$a_{\text{GenDistClass}k}(u; \mathcal{D}_{train}; k; K) =$$

$$= \text{argmax}_{y \in Y} \sum_{i=1}^{|\mathcal{D}_{train}|} [y(x_{(u,i)}) = y] K\left(\frac{\rho(u, x_{(u,i)})}{\rho(u, x_{(u,k+1)})}\right).$$

Выбор (обучение) расстояния

Расстояние можно выбирать.

Пример (взвешенное расстояние Минковского):

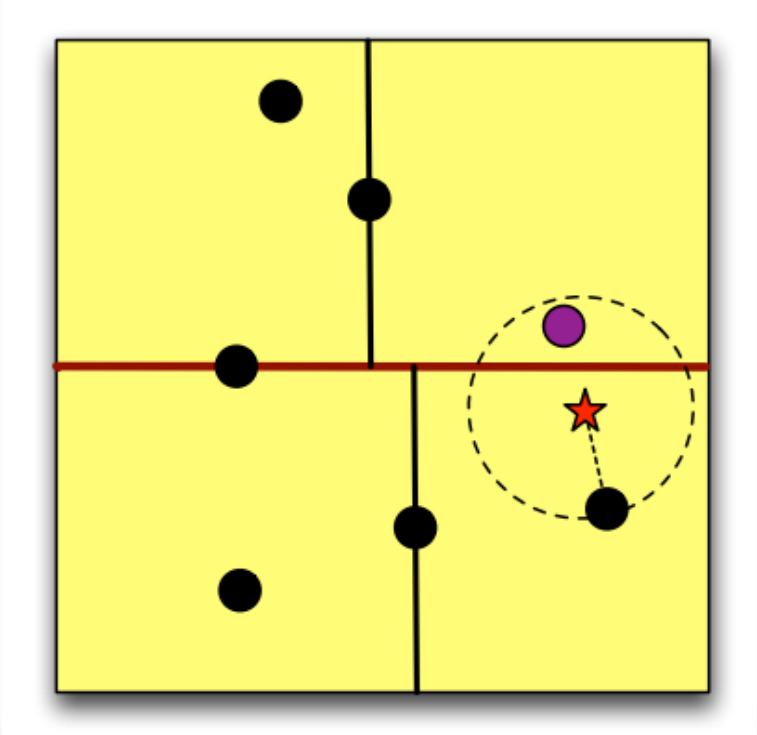
$$\rho(x, y) = \left(\sum_i w_i |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

Обучение этого расстояния — выбор оптимальных знаний w_i.

Выбор ядер также можно отнести к выбору расстояний.

Структуры для хранения данных

Использование структур для хранения данных повышает скорость поиска соседей.
Чаще всего используются ***k-d*-деревья.**



План лекции

- Валидация моделей
- Классификация и регрессия на основе похожести
- Метод одного ближайшего соседа
- Метод *k* ближайших соседей (kNN)
- Обобщенный метрический классификатор
- **Непараметрическая регрессия**
- Оценка качества

Задача регрессии (напоминание)

X — множество объектов, Y — множество меток,

$y: X \to Y$ — неизвестная зависимость, $Y \subseteq \mathbb{R}$

$\mathcal{D} = \{(x_i, y_i)\}$ — обучающее множество.

Задача: найти алгоритм $a: X \to Y$, приближающий y на X.

$a(x) = f(x, \theta)$ — параметрическая функция зависимости, $\theta \in \mathbb{R}^t$.

Основная идея

Предположение: пусть $\theta(x) = \theta$ вокруг $x \in X$:

$$\mathcal{L}(\theta, \mathcal{D}_{train}) = \sum_{i=1}^{|\mathcal{D}_{train}|} w_i(x)(\theta - y_i)^2 \to \min_{\theta \in \mathbb{R}}$$

Основная идея: будем использовать ядерное сглаживание:

$$w_i(x) = K\left(\frac{\rho(x_i, x)}{h}\right),$$

где h — ширина окна.

Формула Надарая-Ватсона

Ядерное сглаживание Надарая-Ватсона:

$$a_{\text{NonParamReg}h}(x, \mathcal{D}_{train}) = \frac{\sum_{x_i \in \mathcal{D}_{train}} y_i w_i(x)}{\sum_{x_i \in \mathcal{D}_{train}} w_i(x)} =$$

$$= \frac{\sum_{x_i \in \mathcal{D}_{train}} y_i K\left(\frac{\rho(x_i, x)}{h}\right)}{\sum_{x_i \in \mathcal{D}_{train}} K\left(\frac{\rho(x_i, x)}{h}\right)}.$$

Основная теорема

Теорема. Если

1) выборка $\mathcal{D}_{train}$ простая, распределенная по $p(x, y)$;

2) $\int_0^\infty K(r)dr < \infty, \lim_{r\to\infty} rK(r) = 0$;

3) $\mathrm{E}(y^2|x) < \infty \; \forall x \in X$;

4) $\lim_{i\to\infty} h_i = 0, \lim_{i\to\infty} ih_i = \infty,$

тогда $a_{\mathrm{NonParamReg}h}(x, \mathcal{D}_{train}) \to^P \mathrm{E}(y|x)$
в любой $x \in X$, где $E(y|x), p(x), D(y|x)$
непрерывны, $p(x) > 0$.

Анализ методов

- выбор функции ядра влияет на гладкость функции потерь;
- выбор функции ядра обычно не столь значим для итогового качества;
- выбор h и k влияет на качество приближения;
- h и k можно настраивать;
- чувствителен к шуму.

План лекции

- Валидация моделей
- Классификация и регрессия на основе похожести
- Метод одного ближайшего соседа
- Метод *k* ближайших соседей (kNN)
- Обобщенный метрический классификатор
- Непараметрическая регрессия
- **Оценка качества**

Матрица ошибок

	Положительный	Отрицательный
Отнесен к положительным	**TP** = True Positive	**FP** = False Positive
Отнесен к отрицательным	**FN** = False Negative	**TN** = True Negative

FP в статистике ошибка первого рода

FNв статистике ошибка второго рода

P = **TP** + **FN** число положительных примеров

N = **FP** + **TN** число отрицательных примеров

Некоторые определения

Полнота (recall) *или* **чувствительность (sensitivity)**:

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{P}}$$

Специфичность (specificity):

$$\text{SPC} = \frac{\text{TN}}{\text{N}}$$

Точность (precision):

$$\text{Precision} = \text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Точность (accuracy):

$$\text{Accuracy} = \text{ACC} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

F-мера

Точность плохо работает для случая несбалансированных классов.

F_β-measure:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

F_1-measure:

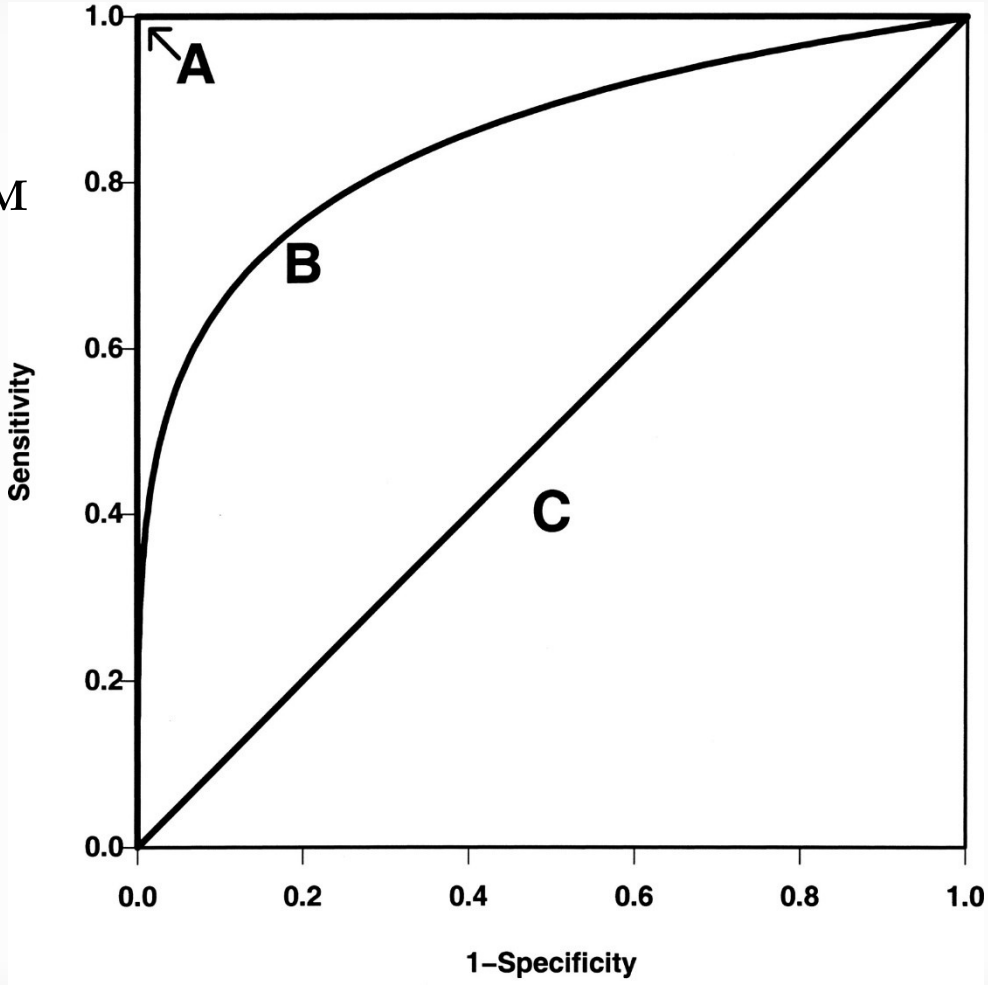
$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROC-кривая

A — лучший алгоритм

B — типичный алгоритм

C — худший алгоритм



AUC

Площадь под кривой (Area under the curve, AUC) это площадь под ROC-кривой.

Случай множества классов

- One vs one классификация
- One vs all (one vs rest) классификация
- Иерархическая классификация
- Матрица ошибок для многих классов

Ошибки для регрессии (1/2)

- Среднеквадратичная ошибка (Mean squared error, MSE):

$$\text{MSE} = \sum_{(x_i, y_i) \in \mathcal{D}} \frac{(a(x_i) - y_i)^2}{|\mathcal{D}|}.$$

- Среднеквадратичная ошибка (Root mean squared error, RMSE):

$$\text{RMSE} = \sqrt{\sum_{(x_i, y_i) \in \mathcal{D}} \frac{(a(x_i) - y_i)^2}{|\mathcal{D}|}}.$$

Ошибки для регрессии (2/2)

- Абсолютная средняя ошибка (Mean absolute error, MAE):

$$\text{MAE} = \sum_{(x_i, y_i) \in \mathcal{D}} \frac{|a(x_i) - y_i|}{|\mathcal{D}|}.$$

- Симметричная средняя абсолютная ошибка в процентах (Symmetric mean absolute percentage error, SMAPE):

$$\text{SMAPE} = \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} \frac{2 \cdot |a(x_i) - y_i|}{|a(x_i)| + |y_i|}.$$

В следующей серии

- Можно ли классифицировать прямолинейно?
- Что такое регуляризация?
- Градиентный спуск