ИТМО

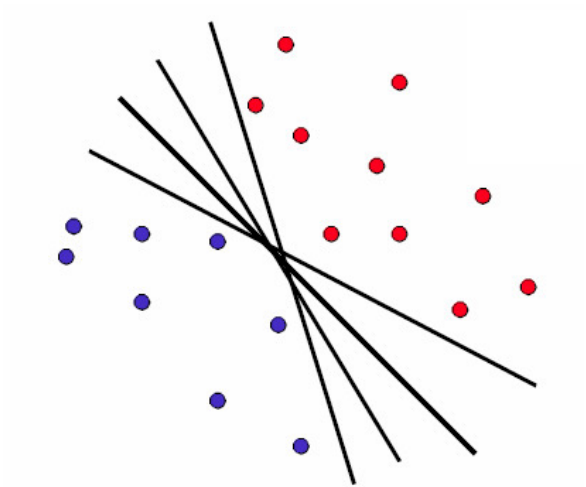
Классификация и категоризация

Техническое зрение

Классификация

Линейный классификатор

- Найдем линейную функцию (гиперплоскость) и разделим положительные $\{y = +1\}$ и отрицательные $\{y = -1\}$ примеры:
  - x_i положительные: $x_i \cdot w + b \geq 0$,
  - x_i отрицательные: $x_i \cdot w + b < 0$.



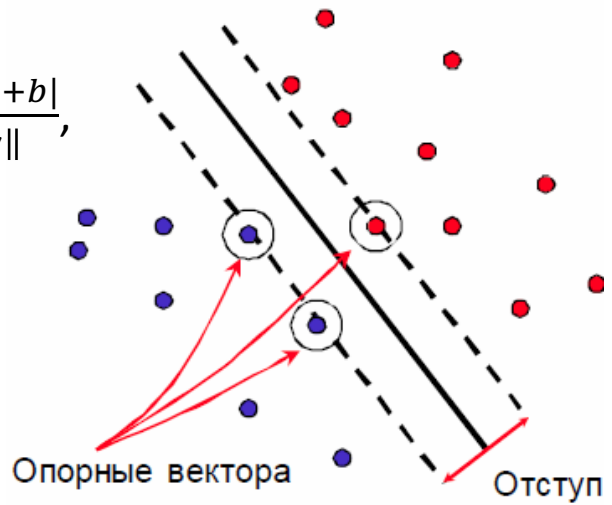
Какая гиперплоскость наилучшая?

Метод опорных векторов

- Support Vector Machine (SVM).
- Базируется на VC-теории.
- *Оптимальной разделяющей гиперплоскостью* называется гиперплоскость, такая, что расстояние от нее до ближайшей точки из обучающей выборки (не важно из какого класса) максимально.
- Оптимальная разделяющая гиперплоскость максимизирует *зазор (отступ)* – расстояние от нее до точек из обучающей выборки.

Метод опорных векторов

- Найдем гиперплоскость, максимизирующую отступ между положительными и отрицательными примерами
  - x_i положительные ($y_i = 1$): $x_i \cdot w + b \geq 1$,
  - x_i отрицательные ($y_i = -1$): $x_i \cdot w + b \leq -1$.
- Для опорных векторов: $x_i \cdot w + b = \pm 1$,
- Расстояние от точки до гиперплоскости равно: $\frac{|x_i \cdot w + b|}{\|w\|}$,
- Отступ равен: $\frac{2}{\|w\|}$.



Метод опорных векторов

- Поиск гиперплоскости:
  1. Максимизируем $\frac{2}{\|w\|}$.
  2. Классифицируем все данные:
     - x_i положительные ($y_i = 1$): $x_i \cdot w + b \geq 1$,
     - x_i отрицательные ($y_i = -1$): $x_i \cdot w + b \leq -1$,
  3. Решим квадратичную оптимизационную задачу:
     - Минимизируем $\frac{1}{2} w^{\mathrm{T}} w$ при условии $y_i(w \cdot x_i + b) \geq 1$.

Метод опорных векторов

- Задача решается при помощи метода множителей Лагранжа:

$$w = \sum_i \alpha_i y_i x_i,$$

где α_i – обученные веса (множители Лагранжа), x_i – опорные вектора.

- Для большей части векторов вес равен нулю.
- Все вектора, для которых вес больше нуля $w > 0$, называются *опорными*.
- Решение будет зависеть только от опорных векторов.

Метод опорных векторов

ИТМО

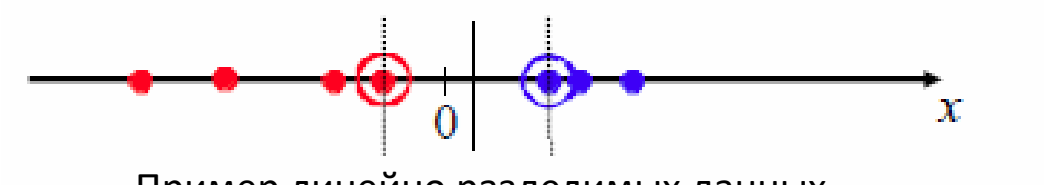
- Решающая функция примет вид:

$$w \cdot x + b = \sum_i \alpha_i y_i x_i \cdot x + b$$

- Решающая функция зависит от скалярных произведений от тестового вектора x и опорных векторов x_i.
- Решение оптимизационной задачи требует вычисления скалярных произведений $x_i \cdot x_j$ между всеми парами векторов из обучающей выборки.

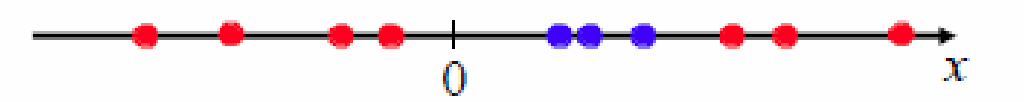
Метод опорных векторов

- На линейно разделимых данных метод опорных векторов работает отлично:



Пример линейно разделимых данных

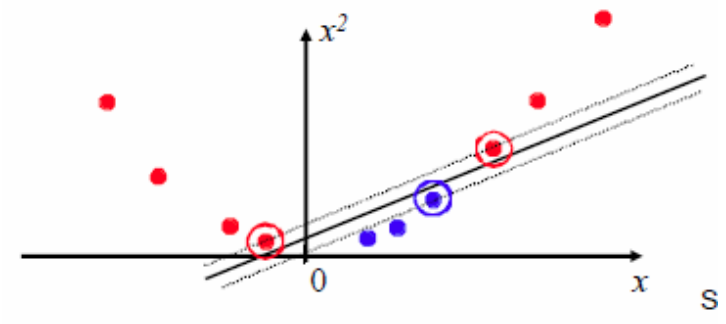
- На более сложных данных метод работает не очень хорошо:



Пример линейно неразделимых данных

Нелинейный метод опорных векторов

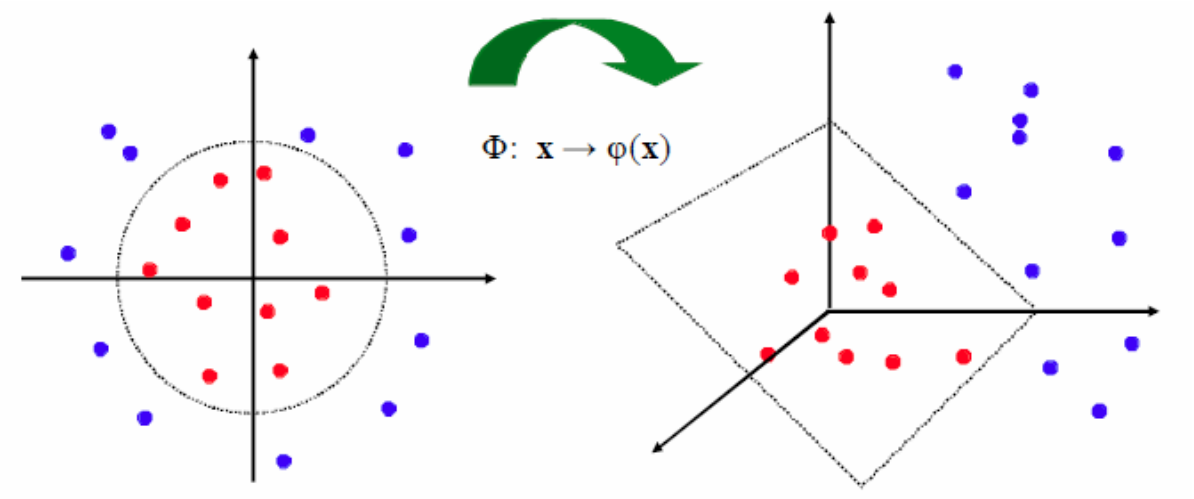
- Основной подход: отображение данных в другое пространство большей размерности и в нем их линейное разделение.



Данные в другом пространстве

Нелинейный метод опорных векторов

- Идея: отражение исходного пространства параметров на какое-то многомерное пространство признаков (feature space), где обучающая выборка линейно разделима:



Нелинейный метод опорных векторов

- Проблема: вычисление скалярных произведений в многомерном пространстве вычислительно сложно.
- Чтобы этого избежать существует подход, который называется *the kernel trick*, сформировавшийся в 1995 году, в котором вместо прямого вычисления преобразования $\varphi(x)$ определяется функция ядра K:

$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$

где матрица $K(x_i, x_j)$ – неотрицательно определённая.

- При помощи данного ядра строится нелинейная решающая функция в исходном пространстве:

$$\sum_i \alpha_i y_i K(x_i, x) + b$$

Пример ядра

- Полиномиальный пример ядер:

$$K(x, y) = ((x, y) + c)^d$$

- Пусть $d = 2, x = (x_1, x_2)$:

$$K(x, y) = \big((x, y) + c\big)^2 = (x_1 y_1 + x_2 y_2 + c)^2 = \varphi(x) \cdot \varphi(y),$$

где:

$$\varphi(x) = \big(x_1^2, x_2^2, \sqrt{2} x_1 x_2, \sqrt{2} c x_1, \sqrt{2} c x_2, c\big),$$

т.е. осуществлен переход из двумерного пространства в шестимерное.

Алгоритм использования SVM

1. Составить обучающую выборку.
2. Выбрать ядро.
3. Вычислить матрицу значений ядра для каждой пары примеров из обучающей выборки.
4. Загрузить матрицу в библиотеку SVM для получения весов и опорных векторов.
5. Во время вывода вычислить значения ядра для тестового образца и каждого опорного вектора и взять взвешенную сумму для получения значения решающей функции.

Многоклассовый SVM

- Нет специальной формулировки SVM для случая многих классов.
- На практике SVM для нескольких классов получается путем комбинации нескольких двухклассовых SVM.
  1. Один против всех:
     - Обучение: обучаем SVM для каждого класса против всех остальных.
     - Вывод: применим все SVM к образцу и назначим класс, SVM для которого выдал наиболее достоверное решение.
  2. Один против одного:
     - Обучение: обучим SVM для каждой пары классов.
     - Вывод: каждый SVM голосует за классы, выбираем класс с наибольшим числом голосов.

Резюме SVM

- **Достоинства:**
  - Много доступных библиотек: http://www.kernel-machines.org/software.
  - Мощный и гибкий подход на основе ядер.
  - На практике работает очень хорошо, даже для маленьких обучающих выборок.
- **Недостатки:**
  - Нет прямых многоклассовых методов, нужно объединять двухклассовые SVM.
  - Ресурсоемкость:
  - При обучении нужно строить полную матрицу ядра для всех примеров.
  - Обучение занимает много времени для больших задач.

Виды ошибок

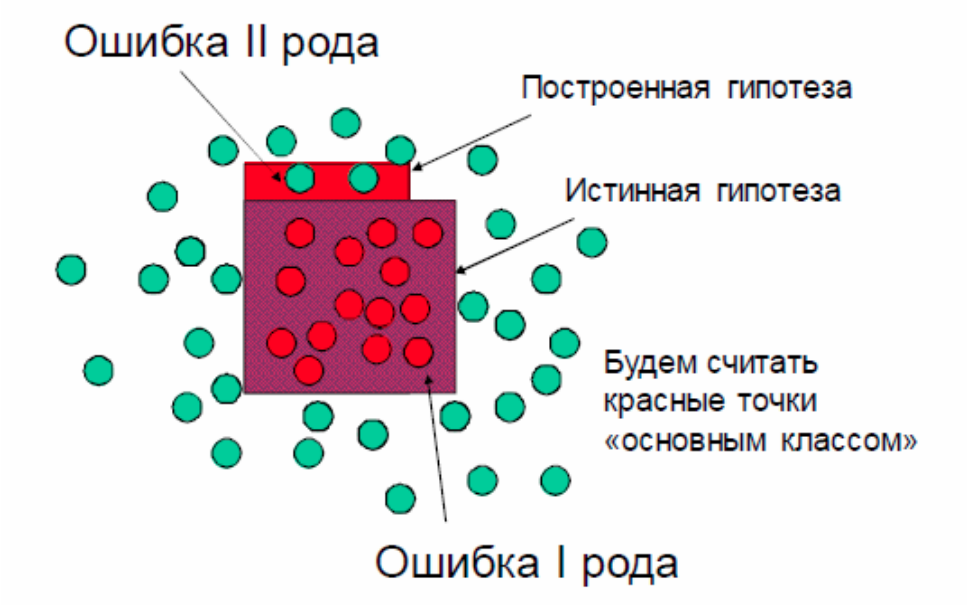
Виды ошибок

- Если рассматривать ошибку как вероятность выдать неверный ответ, то такая мера не является в полной мере достаточной.
  - Например, 15% ошибки при постановке диагноза может означать, как и то что, 15 % больных будут признаны здоровыми (и возможно умрут от отсутствия лечения), так и то, что 15% здоровых больными (и деньги на лечение будут потрачены зря).
- В таких случаях вводят понятие ошибок первого и второго рода, а затем оценивают их по отдельности.

Виды ошибок

- **Пусть существует некоторый основной класс.**
- Как правило, основным классом является класс, при обнаружении которого предпринимается какое-либо действие.
  - В нашем примере при постановке диагноза основным классом будет «болен», а вторичным – «здоров».
- **Ошибкой первого рода** является вероятность принять основной класс за вторичный
  - (вероятность пропуска искомого объекта).
- **Ошибкой второго рода** является вероятность принять вторичный класс за основной
  - (вероятность, когда за искомый объект будет принят фон).

Виды ошибок


Ошибка II рода
Построенная гипотеза
Истинная гипотеза
Будем считать
красные точки
«основным классом»
Ошибка I рода

Виды ошибок

- Особенно важно оценивать ошибки раздельно при сильной несбалансированности классов.
- Например:

$$P(y = +1) = 0{,}01,\ P(y = -1) = 0{,}99.$$

- Тогда при ошибке первого рода $P(f(x) = -1 | y = +1) = 0{,}5$ и нулевой ошибке второго рода общая ошибка будет всего лишь:

$$P(a(x) \neq y) = 0{,}005$$

Виды ошибок

- На основе ошибок можно выделить следующие два параметра:
  - ***Чувствительность*** – вероятность дать правильный ответ на пример основного класса:

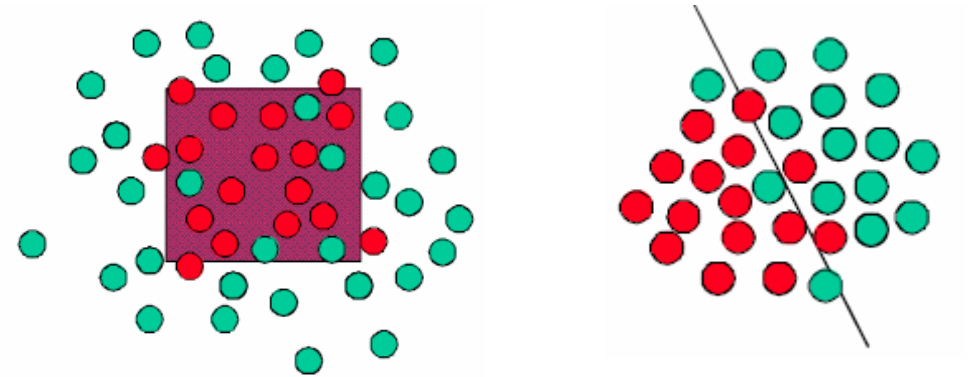
$$sensitivity = P(f(x) = y | y = +1)$$

  - ***Избирательность*** – вероятность дать правильный ответ на пример вторичного класса:

$$specificity = P(f(x) = y | y = -1)$$

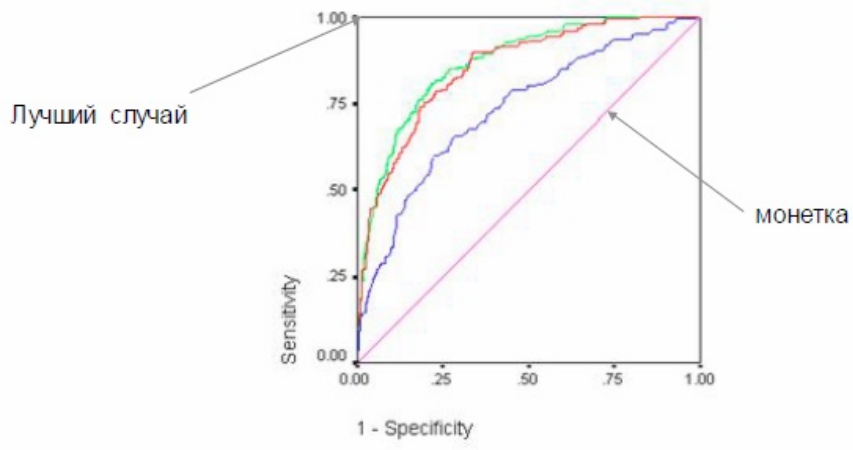
Настройка баланса

- Почти у всех алгоритмов классификации есть параметры, варьируя которые можно получать разные уровни ошибок 1 и 2 рода.



ROC-кривая

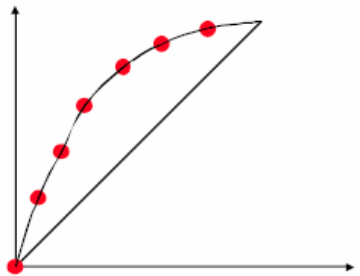
- Для оценки зависимости чувствительности от избирательности можно построить ROC-кривую (Receiver Operating Characteristic curve).



ROC-кривая

1. Для различных значений варьируемого настроечного параметра строится таблица ошибок.
2. Сам параметр в таблице никак не фигурирует,
3. Классификатор строится и оценивается постоянно на разных выборках.
4. Каждой строкой таблицы является точка по которым строится кривая.

Sensitivity	False Positive
0.0	0.0
0.25	0.5
0.5	0.8
…	…
1.0	1.0

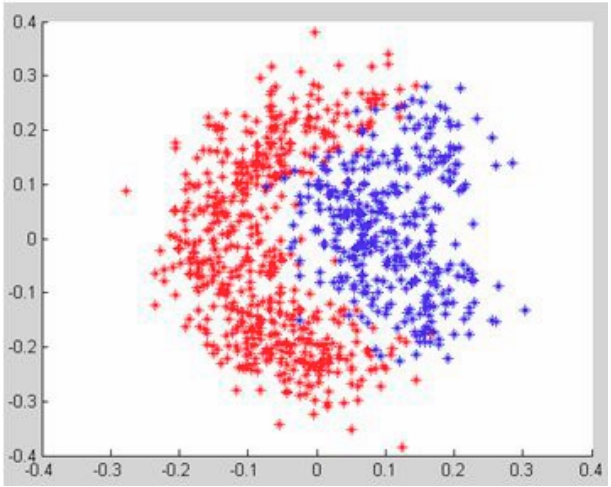


ROC-кривая

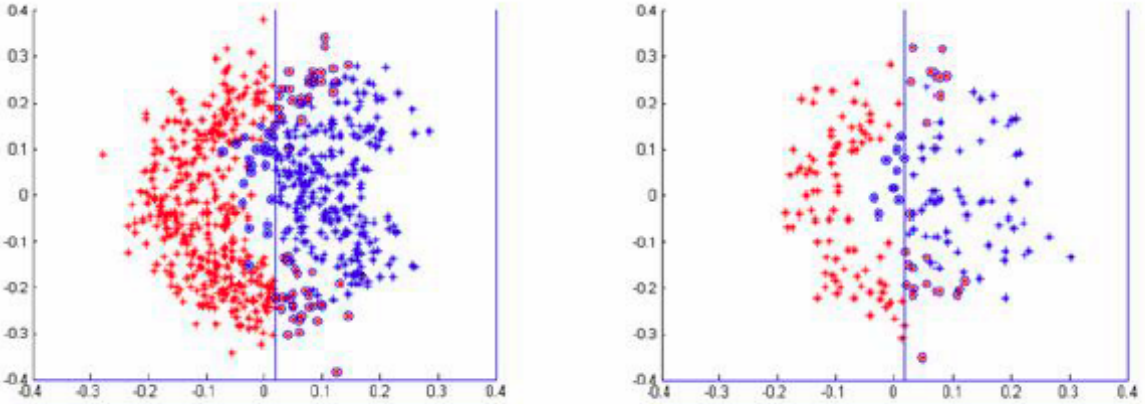
- Площадь под кривой дает некоторый объективный показатель качества классификатора и позволяет сравнивать различные кривые.
- Зачастую для конкретной задачи существуют рамки на ошибку определенного рода. С помощью ROC можно анализировать возможность текущего решения на соответствие требованиям.

Пример

- Даны некоторые точки на плоскости.
- Параметрическим семейством является порог по оси x: $a(x^1, x^2) = \begin{cases} +1, x_1 > \Theta \\ -1, x_2 \leq \Theta \end{cases}$.



Удерживание

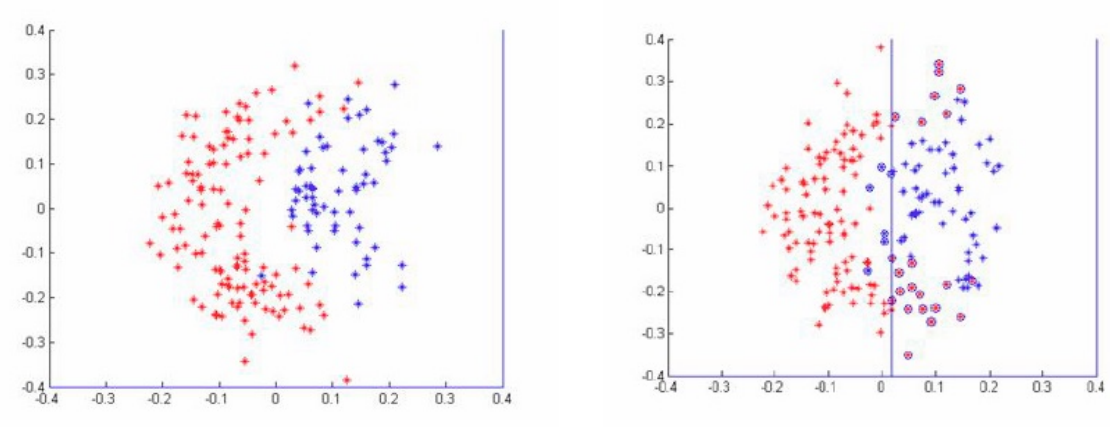


Слева – тренировочная выборка, ошибка 0,1133;
справа – контрольная выборка, ошибка 0,1433

Повторное удерживание

- Тренировочная ошибка:
  - $\{0{,}1097; 0{,}1236; 0{,}1209; 0{,}1250; 0{,}1250\}$,
  - среднее $0{,}1208$.
- Ошибка на контроле
  - $\{0{,}1833; 0{,}1222; 0{,}1333; 0{,}1222; 0{,}1167\}$,
  - среднее $0{,}1356$.

Скользящий контроль



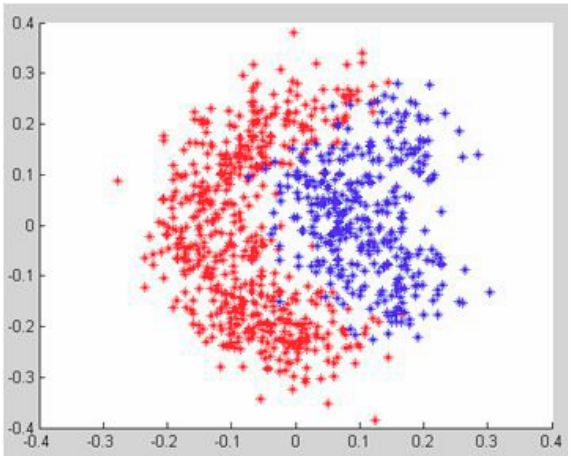
Слева – тренировочная выборка, ошибка 0,1219;
справа – контрольная выборка, ошибка 0,1367

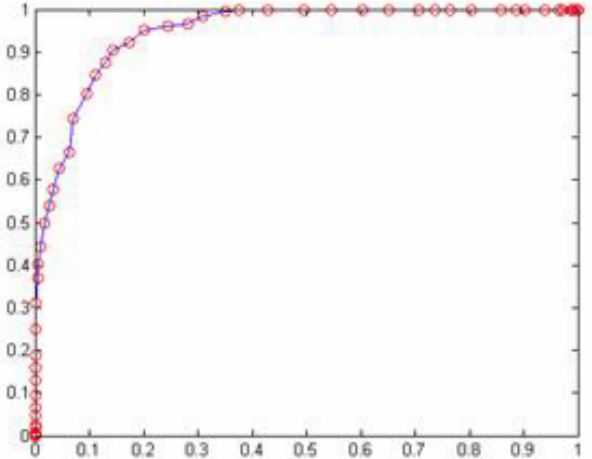
Скользящий контроль

- Тренировочная ошибка:
  - $\{0{,}1236; 0{,}1208; 0{,}1250; 0{,}1097; 0{,}1306\}$,
  - среднее $0{,}1219$.
- Ошибка на контроле
  - $\{0{,}1500; 0{,}1333; 0{,}1222; 0{,}1778; 0{,}1000\}$,
  - среднее $0{,}1367$.

ROC

- Будем менять порог, оценивать ошибку и строить ROC-таблицу, по таблице построим кривую:





ROC

- Общая схема выбора параметров классификатора:
  1. Берём обучающую выборку.
  2. Берём разные параметры классификатора.
     a) Для каждого набора параметров.
        - Используем 5-2 кросс-валидацию.
        - Обучаем классификатор с этими параметрами.
        - Оцениваем ошибки.
     b) Строим ROC-кривую.
  3. Выбираем оптимальные параметры и обучаем классификатор с ними на всей выборке.

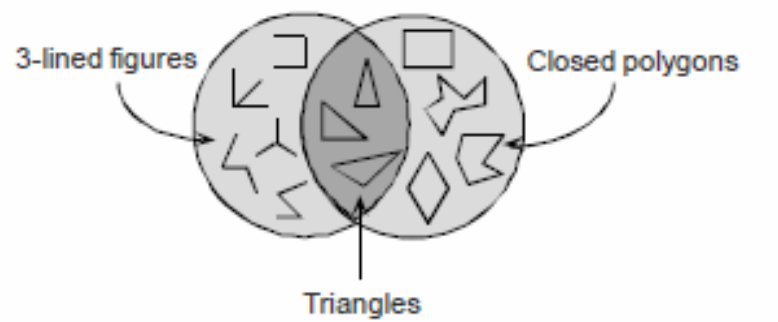
Категоризация изображений

Категоризация изображений

- Многие объекты ведут себя похожим образом.
- Понятие «категории» содержит информацию о том, что мы можем делать с объектом.
- Категорий (слов в языке) меньше, чем объектов в мире.
- Функции объекта зависят от наблюдателя.

Категоризация изображений

- Классическая точка зрения:
  1. Категория определяется набором свойств, общих для всех элементов из категории.
  2. Принадлежность к категории бинарная.
  3. Все элементы из категории одинаковы.
- Пример: Треугольники – это трехсторонние замкнутые ломанные.



Естественные категории

- Определяется наилучшими примерами (*прототипами*).
- Задаем степень соответствия категории.
- Нечеткие правила.

Прототипы и примеры

Модель на прототипе

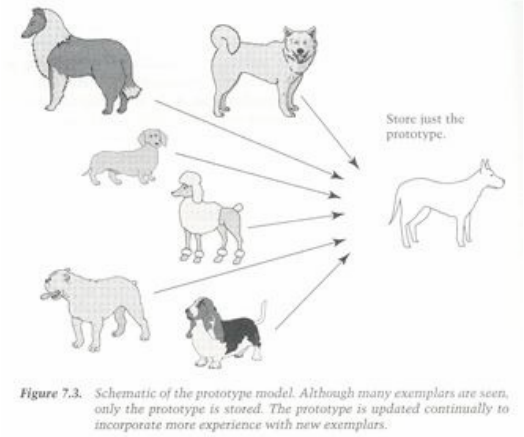


Figure 7.3. Schematic of the prototype model. Although many exemplars are seen, only the prototype is stored. The prototype is updated continually to incorporate more experience with new exemplars.

Решение о соответствии категории принимается путем сравнения объекта с прототипом

Модель на примерах

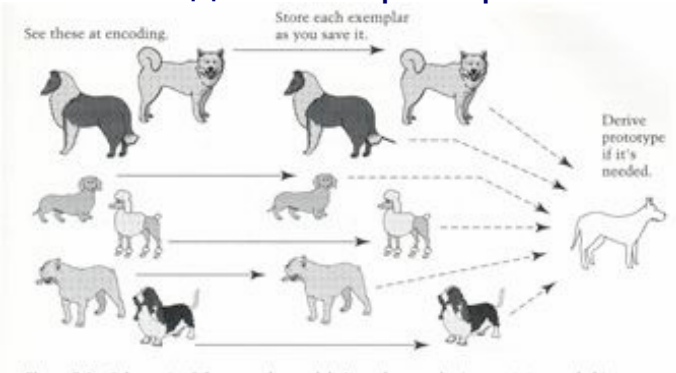


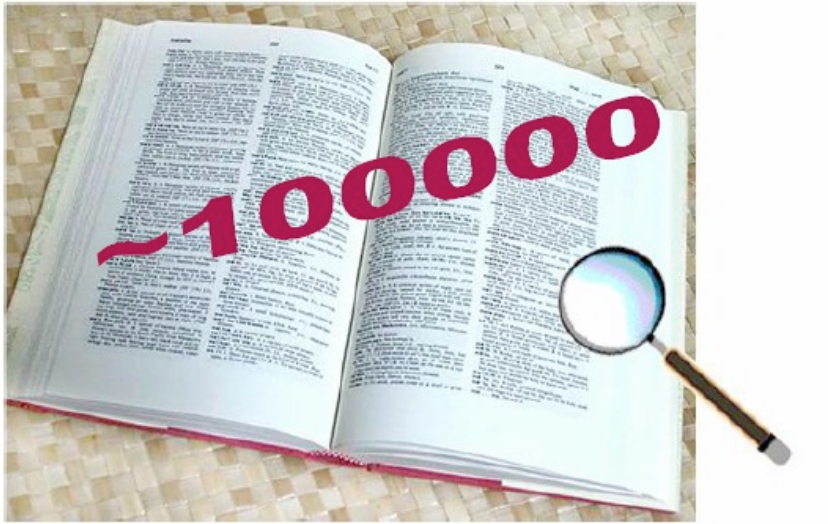
Figure 7.4. Schematic of the exemplar model. As each exemplar is seen, it is encoded into memory. A prototype is abstracted only when it is needed, for example, when a new exemplar must be categorized.

Решение о соответствии категории принимается путем сравнения объекта со множеством примеров из категории

Каноническая перспектива

- Наилучший вид объекта, по которому его проще всего идентифицировать.

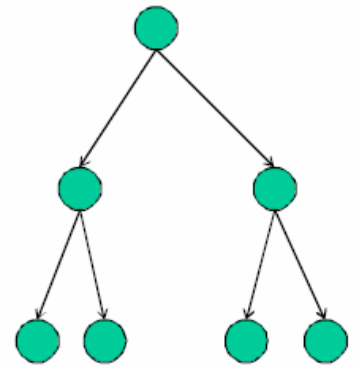
Количество категорий



171000+ слов всего, 47000 устаревших слов, половина существительные

Иерархия категорий

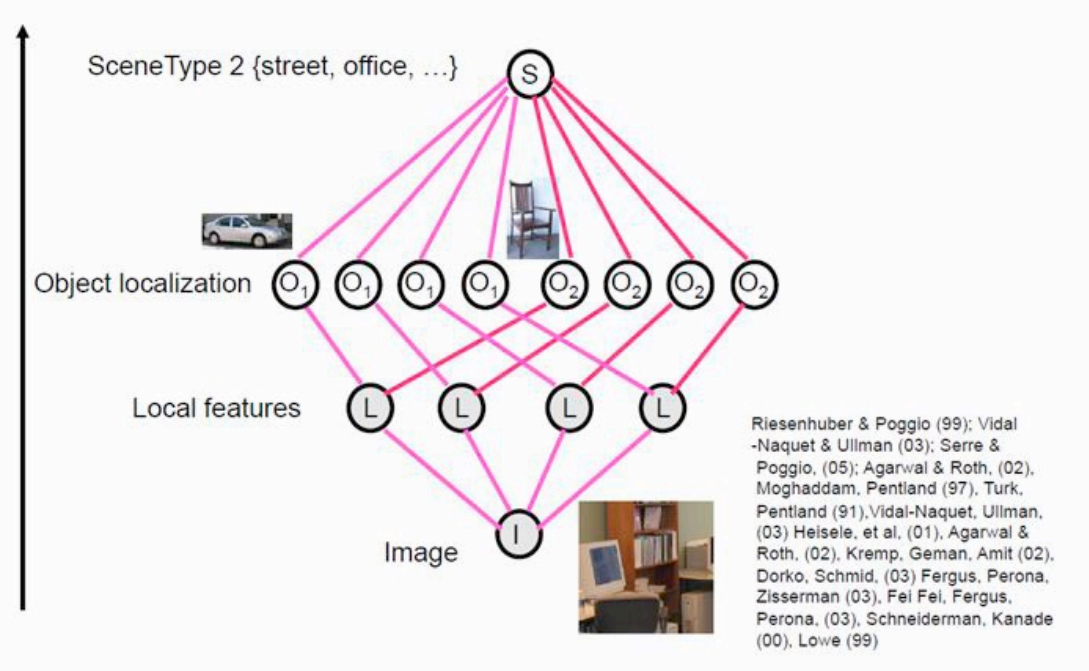
- Можно объединять понятия в группы, выстраивая иерархию категорий:



Задача категоризации изображений

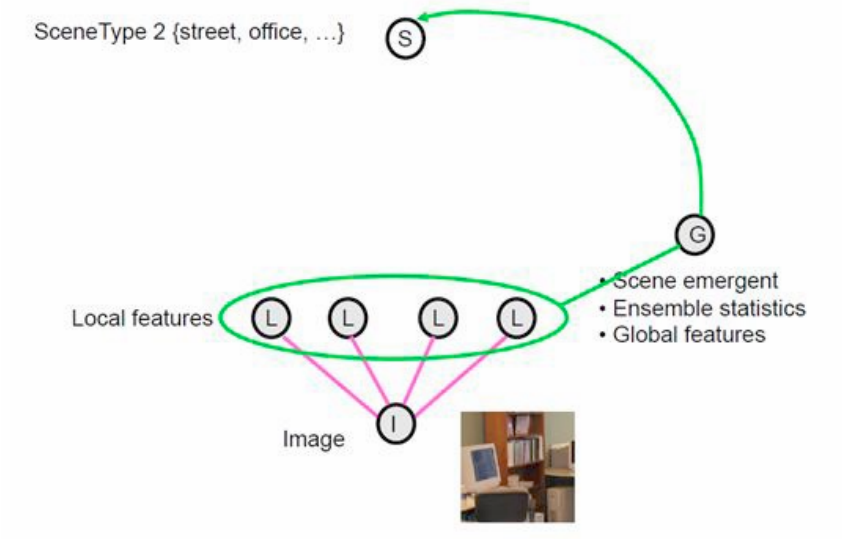
- Определить, есть ли на изображении объект (сцена) заданной категории.

Модель изображения

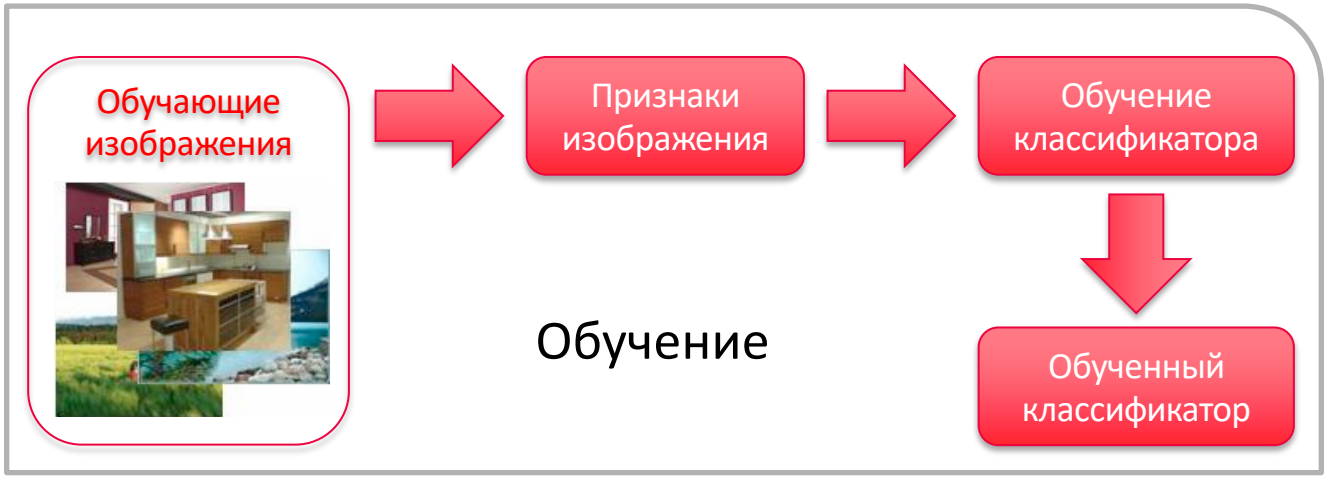


Общий подход к классификации изображений

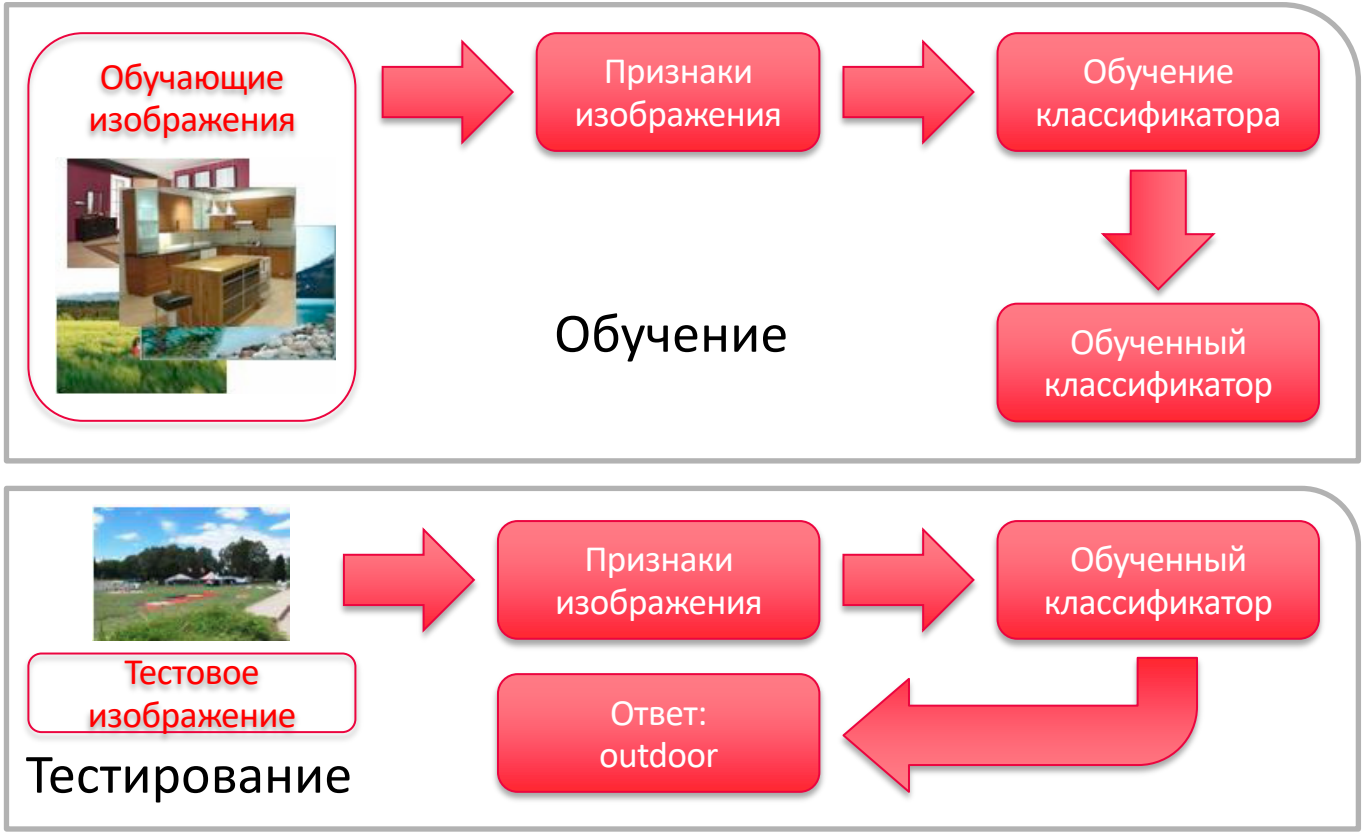
- Анализ только локальных признаков изображения, без выделения и анализа отдельных объектов.



Категоризация



Категоризация

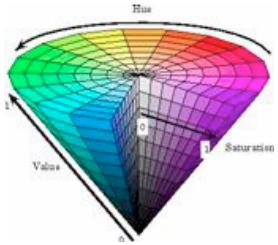


Категоризация

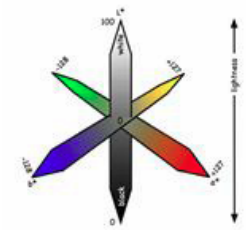
- Будем рассматривать признаки изображения.
- Можно использовать различные методы классификации, часто используется SVM.

Признаки изображения

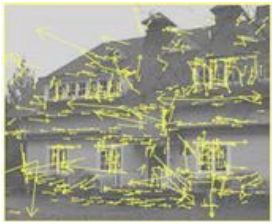
Пространство цветов RGB

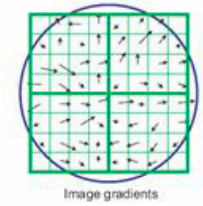


Пространство цветов HSV

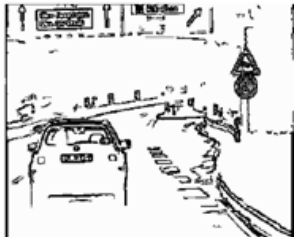


Пространство цветов L*a*b





Градиенты в каждом пикселе



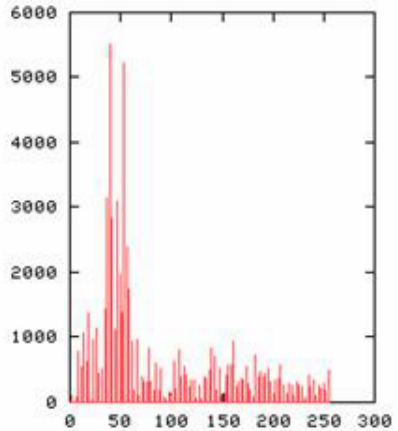
Наличие и ориентация края в каждом пикселе

Использование всех признаков напрямую

- Приведем все изображения к одному размеру.
- Вытянем изображение в вектор, признаки пикселей станут элементами вектора.
- Для распознавания вектор-признак должен быть определенной длины.

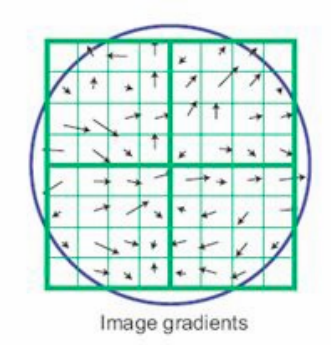
Признаки изображения

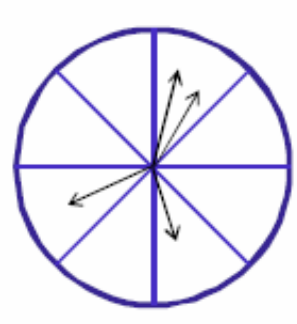
- Можно использовать различные признаки (цвет, края, градиенты, текстуру, и т.д.).
- Использование гистограмм – стандартный способ непараметрического распределения признаков.

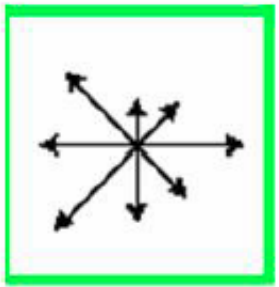


Квантование признаков

- Если разрешение слишком большое, то получается много элементов вектора-признака.
- Чтобы отразить признаки на меньшее разрешение используется квантование:
  - Например, все направления градиента округляются на 45 градусов (8 направлений).







Квантование признаков

- Хотим отличить фотографии тигров от белых медведей, например, по цвету.
- Не оптимально использовать гистограмму цветов на интервале $[0,255]$, поскольку не все цвета встречаются.

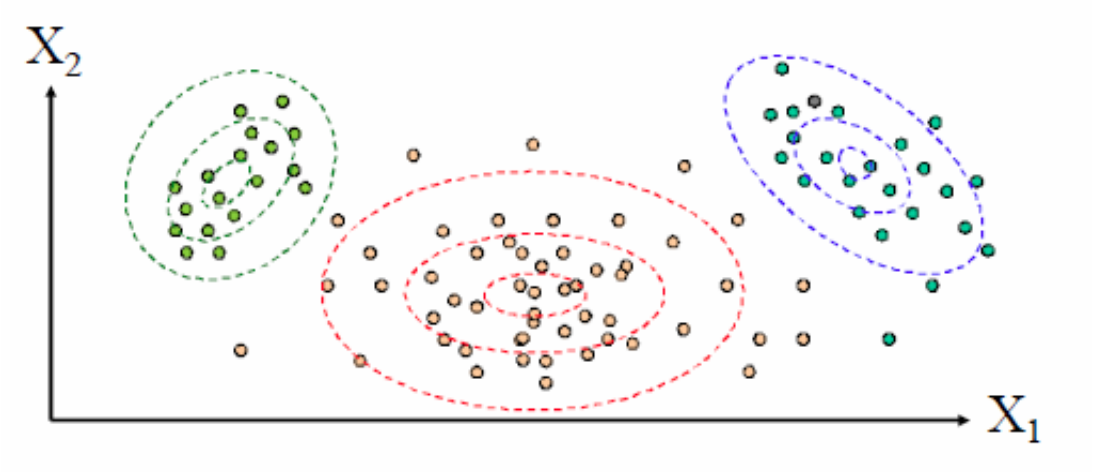
Адаптивное квантование

- Хотим разбить элементы (например, особые пиксели) на часто встречающиеся группы.
- Каждому элементу присвоим номер группы (кластеризация).

Кластеризация

Кластеризация

- Дана обучающая выборка $X_m = \{x_1, \dots, x_m\}, x_i \in R^m$.
- Объекты выборки независимы и взяты из некоторого неизвестного распределения $x_i \in P(x)$.
- Необходимо для всех новых x оценить значения $P(x)$.



Кластеризация методом k-средних

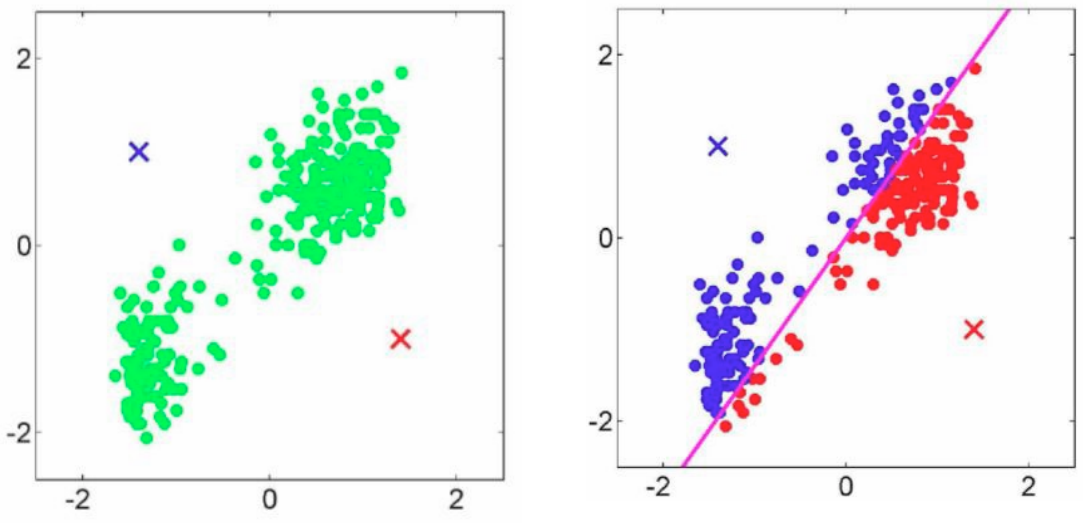
ИТМО

- Минимизируем сумму квадратов Евклидовых расстояний между точками x_i и ближайшими центрами кластеров m_k:

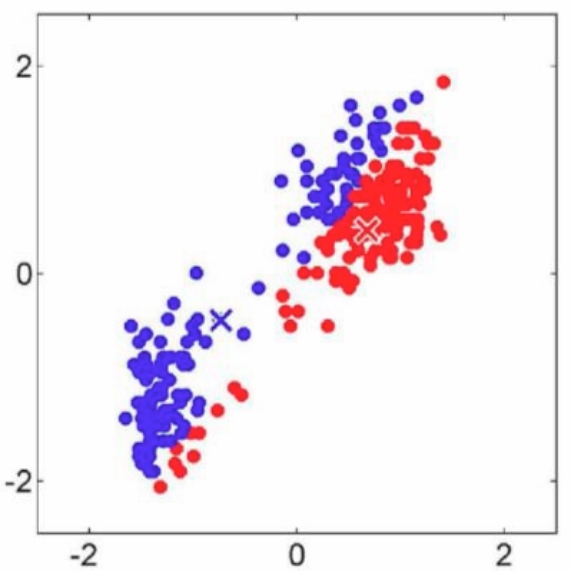
$$D(X, M) = \sum_k \sum_i (x_i - m_k)^2$$

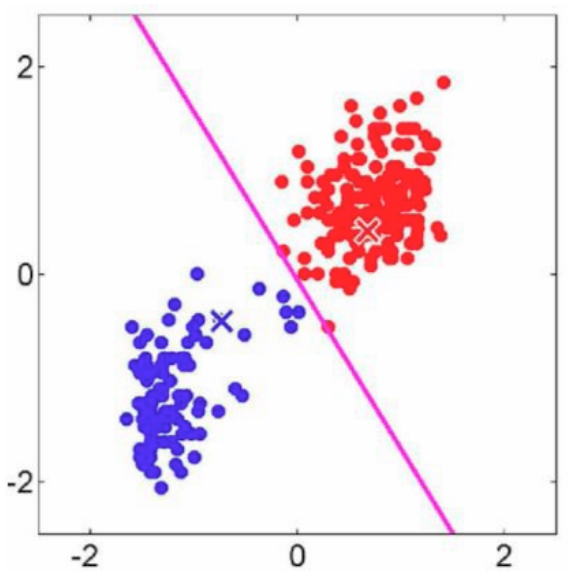
- Алгоритм:
  1. Случайно инициализируем k центров кластеров.
  2. Повторяем до сходимости:
     a. Назначаем каждую точку ближайшему центру;
     b. Пересчитываем центр каждого кластера как среднее всех назначенных точек.

Кластеризация методом k-средних

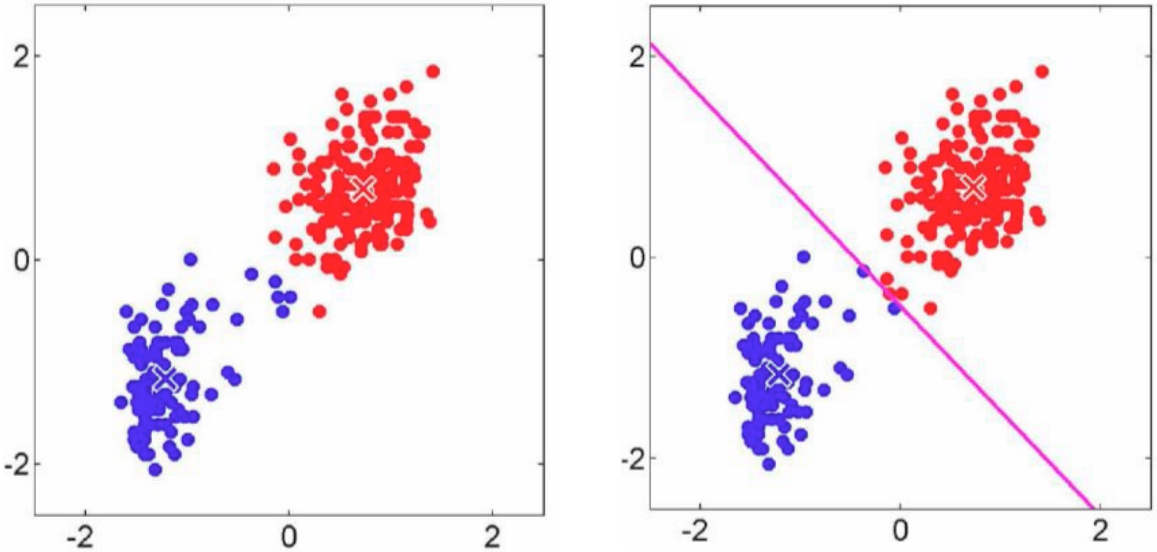


Кластеризация методом k-средних

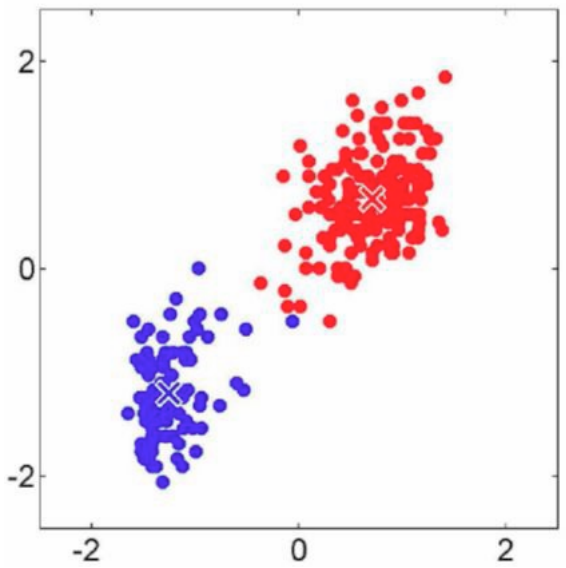




Кластеризация методом k-средних

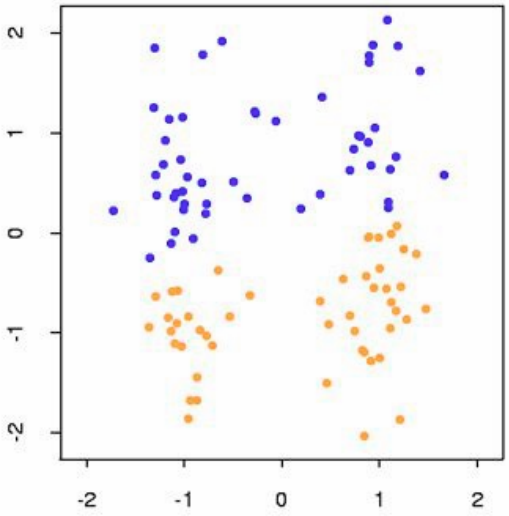


Кластеризация методом k-средних



Кластеризация методом k-средних

- Однопараметрический – требует знания только о количестве кластеров.
- Рандомизирован – зависит от начального приближения.
- Не учитывает строения самих кластеров.



Сравнение гистограмм

1. Пересечение гистограмм (нормализованных):

$$D\left(h_i, h_j\right) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

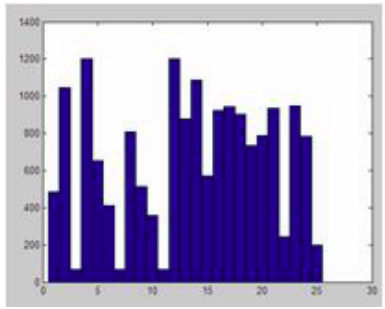
2. Расстояние *L1*:

$$D(h_1, h_2) = \sum_{i=1}^{N} |h_1(i), h_2(i)|$$

3. Расстояние χ^2

$$D(h_1, h_2) = \sum_{i=1}^{N} \frac{\left(h_1(i) - h_2(i)\right)^2}{h_1(i) + h_2(i)}$$

Пространственное распределение



У всех этих трех изображений
похожие гистограммы цветов

Пространственная пирамида изображений

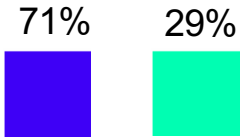
Вычислим гистограмму в каждом блоке и
объединим все гистограммы в один вектор-признак

Общая схема квантования

Вычисление признаков по изображению → Квантуем признаки → Определим область для вычисления → Вычислим статистику по выбранной области

RGB

Квантование на 10 уровней



Посчитанная гистограмма

Вопросы?

ITMO more than a UNIVERSITY

s.shavetov@itmo.ru