

Обработка изображений

**Компиляция презентаций и инфы
из инета**

Вопросы

- Процесс оцифровки изображения. Числовые характеристики изображения. Фильтрация.
- Цветовые модели (монохромная, grayscale, RGB, CMYK, HSV, HLS). Температура изображения.
- Контурный анализ (вектор-контур, НСП, АКФ, ВКФ, свертка)
- Методы выделения контуров на изображении (основанные на вычислении градиента, основанные на поиске нулей).
- Алгоритмы сегментации изображений (волшебная палочка, умные ножницы, разрезы на графах).
- Перевод изображения в монохромное. Нормализация изображений.

Кластеризация изображений:

- 1). Алгоритм k-means.
- 2). Fuzzy c-means.
- 3). Иерархические алгоритмы (дивизимные, агломеративные).
- 4). Послойная кластеризация.
- 5). Алгоритмы поиска минимального остовного дерева.

Выделение связных компонент изображения:

- 1). Рекурсивный алгоритм.
- 2). Итеративный алгоритм.
- 3). Алгоритмы, основанные на границах областей.
- 4). Алгоритмы, основанные на поиске регионов.
- Точечные особенности. Задача слежения.

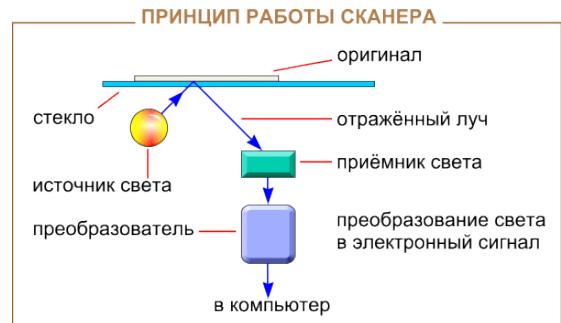
Классификация изображений:

- 1). Наивный байесовский метод.
- 2). Минимизация эмпирического риска.
- Машинное обучение.
- Деревья принятия решения.
- Нейронные сети.
- Метод коррекции ошибок.
- Клика.
- Корреляционные методы анализа изображений.
- Склейка изображений.

Процесс оцифровки изображения

Разрешение и глубина цвета

Оцифровка (англ. digitization) — описание объекта, изображения или аудио- видеосигнала (в аналоговом виде) в виде набора дискретных цифровых замеров (выборок) этого сигнала/объекта, при помощи той или иной аппаратуры, т. е. перевод его в цифровой вид, пригодный для записи на электронные носители.



Разрешение — величина, определяющая количество точек (элементов растрового изображения) на единицу площади (или единицу длины).

Для обозначения разрешающей способности различных процессов преобразования изображений (сканирование, печать, растеризация и т. п.) используют следующие термины:

- **dpi** (англ. dots per inch) — количество точек на дюйм.
- **ppi** (англ. pixels per inch) — количество пикселей на дюйм.
- **lpi** (англ. lines per inch) — количество линий на дюйм, разрешающая способность графических планшетов (дигитайзеров).
- **spi** (англ. samples per inch) — количество сэмплов на дюйм; плотность дискретизации (англ. sampling density), в том числе разрешение сканеров изображений.

По историческим причинам величины стараются приводить к dpi, хотя с практической точки зрения ppi более однозначно характеризует для потребителя процессы печати или сканирования. Измерение в lpi широко используется в полиграфии. Измерение в spi используется для описания внутренних процессов устройств или алгоритмов.

Глубина цвета (качество цветопередачи, битность изображения) — количество бит (объем памяти), используемое для хранения и представления цвета при кодировании одного пикселя растровой графики или видеоизображения.

Часто выражается единицей бит на пикセル (англ. bits per pixel, bpp).

Text
Text

Процесс оцифровки изображения

Оптическая плотность

Оптическая плотность (экстинкция) — мера ослабления света прозрачными объектами (такими, как кристаллы, стекла, фотоплёнка) или отражения света непрозрачными объектами (такими, как фотография, металлы и т. д.).

Вычисляется как десятичный логарифм отношения потока излучения падающего на объект, к потоку излучения прошедшего через него (отразившегося от него), то есть это есть логарифм от величины, обратной к коэффициенту пропускания (отражения).

$$D = \lg \frac{\Phi_{in}}{\Phi_{out}}.$$

К примеру D=4 означает, что свет был ослаблен в $10^4 = 10\,000$ раз, то есть для человека это полностью чёрный объект, а D=0 означает, что свет прошёл (отразился) полностью.

Масштабирование

Масштабирование изображения — изменение размера изображения с сохранением пропорций. Под масштабированием подразумевается как увеличение («апскейлинг» от англ. upscaling), так и уменьшение («даунскейлинг», англ. downscaling) разрешения изображения.

При этом, в зависимости от типа графики (растровая, векторная), масштабирование производится разными алгоритмами. Масштабирование векторных изображений происходит без потерь качества изображения, при увеличении растровых может происходить потеря качества изображения: возможны существенные искажения геометрии мелких деталей и появление ложных узоров на текстурах. Поэтому при масштабировании растровых изображений используются специализированные алгоритмы, сглаживающие нежелательные эффекты.

Дискретизация (Изменение числа точек изображения)

Nearest Neighbor (Метод ближайшего соседа).

Bilinear (Билинейная интерполяция).

Bicubic (Бикубическая интерполяция).

Bicubic Smoother (Бикубический со сглаживанием).

Bicubic Sharper (Бикубический с настройкой резкости).

Числовые характеристики изображения

Статистические моменты

Инварианты

Cartesian moments

$$m_{pq} = \sum_{x=1}^M \sum_{y=1}^N x^p y^q P_{xy}$$

Total mass

$$m_{00} = \sum_{x=1}^M \sum_{y=1}^N P_{xy}$$

Centre Of Mass

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

Centralised moments (инвариантны к переносу)

$$\mu_{pq} = \sum_{x=1}^M \sum_{y=1}^N (x - \bar{x})^p (y - \bar{y})^q P_{xy}$$

Связь центрального момента с обычным

$$\mu_{pq} = \sum_m^p \sum_n^q \binom{p}{m} \binom{q}{n} (-\bar{x})^{(p-m)} (-\bar{y})^{(q-n)} M_{mn}$$

Normalised (инвариантны к переносу и к масштабированию)

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad \gamma = \frac{p+q}{2} + 1 \quad \forall (p+q) \geq 2$$

Hu invariant set (инвариантны к переносу, масштабированию и повороту)

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

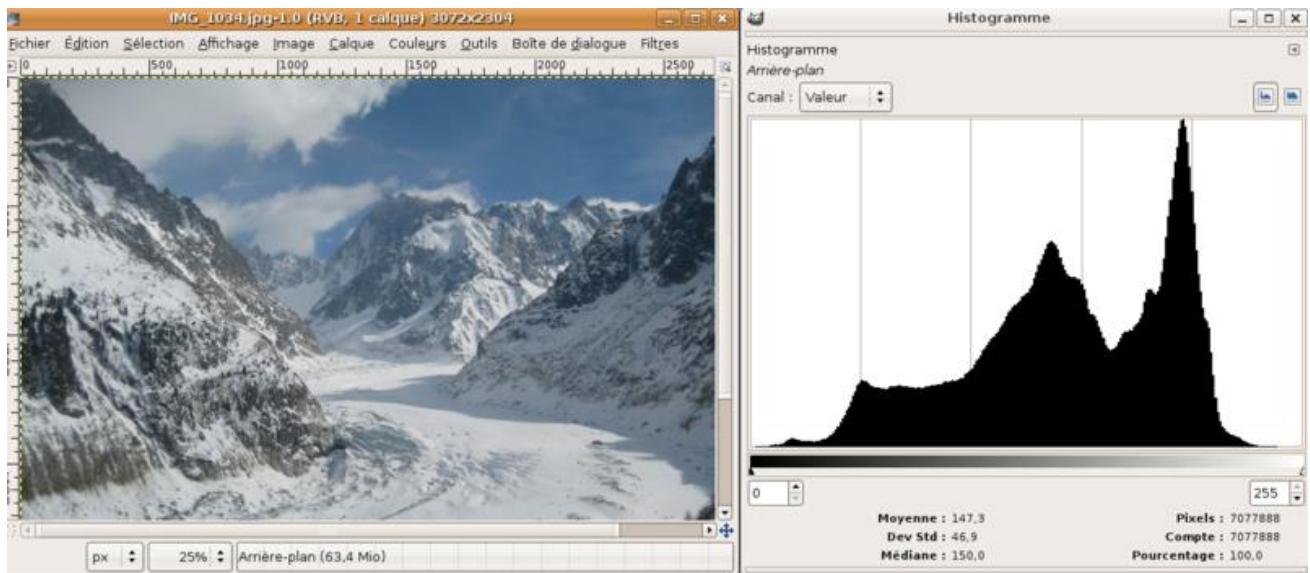
$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].$$

Числовые характеристики изображения

Гистограмма

Гистограмма — это график статистического распределения элементов цифрового изображения с различной яркостью, в котором по горизонтальной оси представлена яркость, а по вертикали — относительное число пикселей с конкретным значением яркости.



Строим массив, заполняем нулями. Обычно массив [0..255]
Цикл, для каждого пикселя:

- Выделяем нужный цветовой канал или находим яркость по формуле. Пиксел -> значение
- Полученное значение должно укладываться в диапазон индексов массива, например [0..255].
- Увеличиваем значение элемента массив[значение] на 1.

Конец цикла.

Полученный массив и представляет собой гистограмму, элементы массива — означают высоты столбиков.

Гистограмма инвариантна к переносу и повороту

Минус – игнорирует пространственное расположение и текстуру объектов на изображении (выход: можно предварительно сегментировать изображение на блоки).

Есть еще гистограмма цвета – по аналогии, она отражает количество пикселей для каждого цвета изображения. Можно получить из гистограмм «цветных» каналов – R, G и B, например.

Числовые характеристики изображения

Признаки текстуры

Текстуру в том числе определяет пространственное распределение серых значений, позволяющее оценить признаки изображения, связанные со статическими вычислениями второго порядка. Наиболее известна **матрица смежности** уровня серого Харалика (или матрица яркостной зависимости, Gray-Level Co-occurrence Matrix).

Построение матрицы P размером $L \times L$ для L квантованных значений яркости текстуры изображения с матрицей яркости I выполняется по следующему правилу:

$$P_{r,\theta}(i, j) = |\{(k, s), (t, v) : I(k, s) = i, I(t, v) = j\}|,$$

где

- (1) i, j — уровни яркости матрицы $P(i, j = \overline{1, L})$;
- (2) $I(k, s)$ и $I(t, v)$ — значения элементов матрицы яркости с координатами (k, s) и (t, v) ;
- (3) r — расстояние между элементами $I(k, s)$ и $I(t, v)$;
- (4) θ — угол между элементами $I(k, s)$ и $I(t, v)$ относительно горизонтальной оси, т.е. направление [34, 35].

Ковариационные матрицы описывают пространственные связи пар яркостей элементов текстуры (согласно предположению Юлеша, мозг человека для распознавания текстур использует статистики лишь первого и второго порядка). Для фиксированных расстояния и угла рассчитываются 14 признаков: второй угловой момент, контрастность, энтропия, корреляция и др. (см. Табл. 1)

Таблица 1. Основные текстурные признаки, рассчитанные по матрице зависимости

№	Тектурный признак	Формула вычисления
1	Второй угловой момент	$\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} P(i, j)^2$
2	Контрастность	$\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} (i - j)^2 P(i, j)$
3	Энтропия	$-\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} P(i, j) \log_2 P(i, j)$
4	Корреляция	$\frac{\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} ij P(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y}$

$$\mu_x = \sum_{i=0}^{L-1} iP(i, x), \quad \mu_y = \sum_{j=0}^{L-1} jP(y, j),$$
$$\sigma_x^2 = \sum_{i=0}^{L-1} (i - \mu_x)^2 P(i, x), \quad \sigma_y^2 = \sum_{j=0}^{L-1} (j - \mu_y)^2 P(y, j),$$

$$P(i, x) = \sum_{j=0}^{L-1} P(i, j), \quad P(y, j) = \sum_{i=0}^{L-1} P(i, j).$$

Числовые характеристики изображения

Построение матрицы смежности. Пример



Рис. 8. Фрагмент исходного изображения.

Запишем значения яркости пикселей фрагмента:

$$\begin{pmatrix} 123 & 32 & 68 & 245 \\ 7 & 96 & 38 & 121 \\ 67 & 200 & 48 & 72 \\ 10 & 54 & 160 & 201 \end{pmatrix}.$$

Далее составим матрицу квантовых яркостей. Если $I_{P(x,y)} < 50$, то соответствующий элемент матрицы квантовых яркостей будет равен 0; если, $50 \leq I_{P(x,y)} < 150$, то элемент будет равен 1; если

$150 \leq I_{P(x,y)} < 200$, то 2; если $I_{P(x,y)} \geq 200$, то 3: $\begin{pmatrix} 1 & 0 & 1 & 3 \\ 0 & 1 & 0 & 1 \\ 1 & 3 & 0 & 1 \\ 0 & 0 & 2 & 3 \end{pmatrix}$.

Общий вид матрицы смежности:

$$\begin{pmatrix} \#(0,0) & \#(0,1) & \#(0,2) & \#(0,3) \\ \#(1,0) & \#(1,1) & \#(1,2) & \#(1,3) \\ \#(2,0) & \#(2,1) & \#(2,2) & \#(2,3) \\ \#(3,0) & \#(3,1) & \#(3,2) & \#(3,3) \end{pmatrix}.$$

Матрицы смежности для угловых направлений $0^\circ, 45^\circ, 90^\circ$ и 135° , содержат в себе информацию о количестве смежных сочетаний элементов матрицы квантовых яркостей.

Числовые характеристики изображения

Построение матрицы смежности. Пример

$$P_{0^\circ} = \begin{pmatrix} 2 & 6 & 1 & 1 \\ 6 & 0 & 0 & 2 \\ 1 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}; P_{45^\circ} = \begin{pmatrix} 4 & 1 & 0 & 3 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{pmatrix};$$

$$P_{90^\circ} = \begin{pmatrix} 2 & 5 & 1 & 1 \\ 5 & 2 & 0 & 3 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \end{pmatrix}; P_{135^\circ} = \begin{pmatrix} 2 & 3 & 0 & 2 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{pmatrix}.$$

Полученные матрицы покажут наличие текстуры на фрагменте исследуемого изображения. В нашем случае, фрагмент представляет собой выделенное зерно.

В случае если на одной из 4-х матриц смежности будет прослеживаться ориентированный в этом направлении перепад яркости, то можно будет говорить о преимущественной ориентации полос скольжения в зерне, по соответствующему направлению.

(тему с зерном игнорируем)

Числовые характеристики изображения

Фильтр Габора

Фильтр Габора — линейный электронный фильтр, импульсная переходная характеристика которого определяется в виде гармонической функции, помноженной на гауссиан. При цифровой обработке изображений этот фильтр применяется для распознавания границ объектов.

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$
$$x' = x \cos \theta + y \sin \theta \quad y' = -x \sin \theta + y \cos \theta$$

Обработка изображения фильтром Габора достигается путём усреднения значений обрабатываемого изображения по некоторой области в каждой точке. Соответственно, наложение фильтра Габора на изображение имеет вид:

$$I'(x, y) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n I\left(x - \frac{n}{2} + i, y - \frac{n}{2} + j\right) \cdot G(i, j)$$

где:

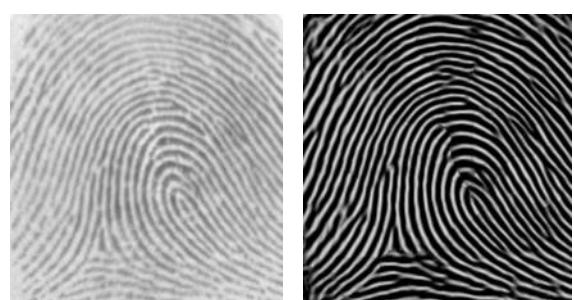
$I(x, y)$ - интенсивность исходного изображения в точке (x, y) ,

$I'(x, y)$ - интенсивность нового изображения в точке (x, y) ,

$G(i, j)$ - значение функции Габора, $i \in [0, n]$, $j \in [0, n]$.

Если отбросить синусоидальную составляющую функции в фильтре Габора, он выродится в фильтр Гауссова размытия (Gaussian Blur). Поэтому очевидно, что эти два фильтра имеют практически одинаковый алгоритм применения, различающийся в некоторых деталях.

(Обычно используются серии фильтров с различными фи (поворот ядра Габора)).



Числовые характеристики изображения

Признаки формы: дескрипторы Фурье

Введение. Распознавание образов является главной задачей в области машинного зрения. Многие задачи распознавания объектов на изображениях могут сводиться к распознаванию фигур – частному случаю распознавания образов. Эффективным и легкореализуемым способом представления фигур является использование фурье-дескрипторов [1]. В данной работе фурье-дескрипторы применяются совместно с нейронной сетью для решения задачи распознавания фигур.

Функция расстояния. Функция расстояния R_n для контура $P_n = (x_n, y_n)$, $n=[1, N]$ вычисляется как расстояние от неподвижной точки $C(x_0, y_0)$ до каждой точки (x_n, y_n) . В качестве точки C обычно выбирается центроид фигуры [2]:

$$R_n = \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2}.$$

Функция расстояния имеет те же преимущества и недостатки, что и комплексные координаты.

Касательный угол. Каждый контур считается кривой линией, поэтому можно рассчитать угол наклона прямой, касательной к каждой его точке [2]:

$$\theta_n = \operatorname{arctg} \left(\frac{y_n - y_{n-w}}{x_n - x_{n-w}} \right).$$

Здесь w – окно небольшого размера.

Несмотря на простоту реализации, данный метод имеет два существенных недостатка: чувствительность к шуму и прерывность. Для того чтобы избежать прерывности, определяется кумулятивная угловая функция $\varphi_n = \theta_n - \theta_0$, где θ_0 – касательный угол к случайной выбранной точке на контуре. До расчета этой функции часто применяется фильтр низких частот. В настоящей работе кумулятивная угловая функция используется в качестве исходной функции для фурье-преобразования.

Фурье-дескрипторы. Фурье-дескрипторы получаются в результате применения фурье-преобразования к указанным выше одномерным функциям представления фигуры [3, 4]. Фурье-дескрипторами называются нормированные коэффициенты фурье-разложения. Предположим, что контур объекта обозначается непрерывной и периодичной функцией $c(t)$, при этом

$$a_k = \frac{2}{T} \int_0^T c(t) \cos(k\omega t) dt, \quad b_k = \frac{2}{T} \int_0^T c(t) \sin(k\omega t) dt, \quad c_k = \sqrt{a_k^2 + b_k^2}$$

(a_k – реальная часть; b_k – мнимая часть; c_k – фурье-дескриптор).

Фурье-дескрипторы устойчивы к перемещению, масштабированию и вращению объекта [2, 3], что позволяет использовать их для представления фигуры.

Числовые характеристики изображения

Дескрипторы Фурье

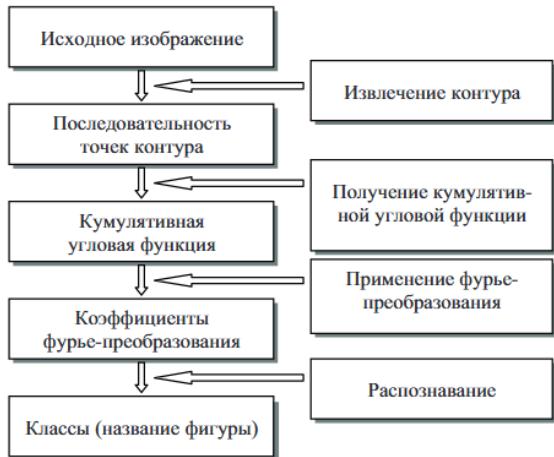


Рис. 1. Общая схема алгоритма

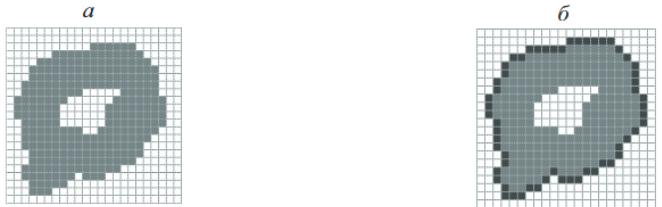


Рис. 3. Выделение контура с использованием алгоритма Мора:
а – исходное изображение; б – объект с выделенным контуром

Выделенный контур сохраняется в виде массива точек $P_n = (x_n, y_n)$, $n = [1, N]$, где N – количество граничных точек. Точки упорядочены по часовой стрелке. В каждой точке определяется угол наклона касательной линии к горизонтальной оси (угловая функция). Угловая функция меняется в диапазоне $[0, 2\pi]$. Таким образом, она прерывна (резкий переход из 2π в 0) и не может служить исходной функцией для фурье-преобразования. Для устранения этой проблемы используется кумулятивная угловая функция. Однако кумулятивная функция имеет ряд недостатков: она прерывна в последней точке контура и ее значения зависят от длины контура. Для того чтобы можно было применить фурье-преобразование, кумулятивная функция должна быть нормирована [3]:

$$\varphi^*(t) = \varphi\left(\frac{L}{2\pi}t\right) + t$$

(φ – кумулятивная функция; φ^* – нормированная функция; L – длина контура).

В результате применения фурье-преобразования к нормированной кумулятивной угловой функции имеем (рис. 4, д–жс):

$$a_k = \frac{1}{\pi} \int_0^{2\pi} \varphi^*(t) \cos(kt) dt, \quad b_k = \frac{1}{\pi} \int_0^{2\pi} \varphi^*(t) \sin(kt) dt, \quad c_k = \sqrt{a_k^2 + b_k^2}.$$

Полученные таким образом фурье-дескрипторы инвариантны к перемещению, масштабированию и вращению и могут быть использованы как входные данные для нейронной сети. Количество коэффициентов фурье-преобразования для нейронной сети будет зависеть от "сложности" фигуры. Эксперимент показывает, что для распознавания несложных тригонометрических фигур достаточно 15–20 коэффициентов. В данной работе используются 20 коэффициентов (дескрипторов).

Числовые характеристики изображения

Локальные признаки: особые точки

Особая точка m , или точечная особенность (англ. point feature, key point, feature), изображения – это точка изображения, окрестность которой $\text{o}(m)$ можно отличить от окрестности любой другой точки изображения $\text{o}(n)$ в некоторой другой окрестности особой точки $\text{o}_2(m)$. В качестве окрестности точки изображения для большинства алгоритмов берётся прямоугольное окно, составляющее размер 5×5 пикселей. Процесс определения особых точек достигается путём использования детектора и дескриптора.

Детектор – это метод извлечения особых точек из изображения. Детектор обеспечивает инвариантность нахождения одних и тех же особых точек относительно преобразований изображений.

Дескриптор – идентификатор особой точки, выделяющий её из остального множества особых точек. В свою очередь, дескрипторы должны обеспечивать инвариантность нахождения соответствия между особыми точками относительно преобразований изображений.

Свойства особых точек

- **Отличимость** (distinctness) – особая точка должна явно выделяться на фоне и быть отличимой (уникальной) в своей окрестности.
- **Инвариантность** (invariance) – определение особой точки должно быть независимо к аффинным преобразованиям.
- **Стабильность** (stability) – определение особой точки должно быть устойчиво к шумам и ошибкам.
- **Уникальность** (uniqueness) – кроме локальной отличимости, особая точка должна обладать глобальной уникальностью для улучшения различимости повторяющихся паттернов.
- **Интерпретируемость** (interpretability) – особые точки должны определяться так, чтобы их можно было использовать для анализа соответствий и выявления интерпретируемой информации из изображения.
- **Повторяемость** (repeatability) – особая точка находится в одном и том же месте сцены или объекта изображения, несмотря на изменения точки обзора и освещённости.

Числовые характеристики изображения

Локальные признаки: особые точки

Свойства особых точек

- **Отличительность** / информативность (distinctiveness/informativeness) – окрестности особых точек должны иметь большие отличия друг от друга, так, чтобы возможно было выделить и сопоставить особые точки.
- **Локальность** (locality) – особая точка должна занимать небольшую область изображения, чтобы быть уменьшить вероятность чувствительности к геометрическим и фотометрическимискажениям между двумя изображениями, снятых в различных точках обзора.
- **Количество** (quantity) – число обнаруженных особых точек должно быть достаточно большим, так чтобы их хватило для обнаружения даже небольших объектов. Однако оптимальное количество особых точек зависит от предметной области. В идеале количество обнаруженных особых точек должно адаптивно определяться с использованием простого и интуитивного порога. Плотность расположения особых точек должна отражать информационное содержимое изображения, чтобы обеспечить его компактное представление.
- **Точность** (accuracy) – обнаруженные особые точки должны точно локализовываться, как в исходном изображении, так и взятом в другом масштабе.
- **Эффективность** (efficiency) – время обнаружения особых точек на изображении должно быть допустимым в критичных по времени приложениях.

Углы (corners) – особые точки, которые формируются из двух или более граней, и грани обычно определяют границу между различными объектами и / или частями одного и того же объекта. По-другому можно сказать, что углы – это точки, у которых в окрестности интенсивность изменяется относительно центра (x,y). Углы определяются по координатам и изменениям яркости окрестных точек изображения. Главное свойство таких точек заключается в том, что в области вокруг угла у градиента изображения преобладают два доминирующих направления, что делает их различимыми.

Числовые характеристики изображения

Локальные признаки: особые точки

Свойства особых точек

- **Отличительность** / информативность (distinctiveness/informativeness) – окрестности особых точек должны иметь большие отличия друг от друга, так, чтобы возможно было выделить и сопоставить особые точки.
- **Локальность** (locality) – особая точка должна занимать небольшую область изображения, чтобы быть уменьшить вероятность чувствительности к геометрическим и фотометрическимискажениям между двумя изображениями, снятых в различных точках обзора.
- **Количество** (quantity) – число обнаруженных особых точек должно быть достаточно большим, так чтобы их хватило для обнаружения даже небольших объектов. Однако оптимальное количество особых точек зависит от предметной области. В идеале количество обнаруженных особых точек должно адаптивно определяться с использованием простого и интуитивного порога. Плотность расположения особых точек должна отражать информационное содержимое изображения, чтобы обеспечить его компактное представление.
- **Точность** (accuracy) – обнаруженные особые точки должны точно локализовываться, как в исходном изображении, так и взятом в другом масштабе.
- **Эффективность** (efficiency) – время обнаружения особых точек на изображении должно быть допустимым в критичных по времени приложениях.

Углы (corners) – особые точки, которые формируются из двух или более граней, и грани обычно определяют границу между различными объектами и / или частями одного и того же объекта. По-другому можно сказать, что углы – это точки, у которых в окрестности интенсивность изменяется относительно центра (x,y). Углы определяются по координатам и изменениям яркости окрестных точек изображения. Главное свойство таких точек заключается в том, что в области вокруг угла у градиента изображения преобладают два доминирующих направления, что делает их различимыми.

Числовые характеристики изображения

Локальные признаки: особые точки

Детектор углов Моравес

Работа в исследовании привязки изображений с использованием особых точек началась детектора Моравеца (Moravec, 1977). Детектор Моравеца – самый простой из существующих. Автор рассматривает изменение яркости квадратного окна W (обычно размера 3x3, 5x5, 7x7 пикселей) относительно интересующей точки при сдвиге окна W на 1 пиксель в 8-ми направлениях (горизонтальных, вертикальных и диагональных) [4]. Алгоритм:

1. Для каждого пикселя (x,y) в изображении вычислить изменение интенсивности

$$V_{u,v}(x,y) = \sum_{\forall u,v \in w} (I(x+u+a, y+v+b) - I(x+a, y+b))^2,$$
$$(u,v) \in \{(1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1)\}$$

2. Построить карту вероятности нахождения углов в каждом пикселе (x,y) изображения посредством вычисления оценочной функции $C(x,y) = \min(V_{u,v}(x,y))$. То есть определяется направление, которому соответствует наименьшее изменение интенсивности, т.к. угол должен иметь смежные ребра.
3. Отсечь пиксели, в которых значения $C(x,y)$ ниже порогового значения T .
4. Удалить повторяющиеся углы с помощью применения процедуры поиска локальных максимумов функции отклика (non-maximal suppression). Все полученные ненулевые элементы карты соответствуют углам на изображении.

Детектор Моравеца обладает свойством анизотропии в 8 направлениях смещения окна. Основными недостатками рассматриваемого детектора являются отсутствие инвариантности к преобразованию поворота и возникновение ошибок детектирования при наличии большого количества диагональных ребер [2].

В работах [46, 47, 50] показано, что в наибольшей степени требованием сегментации соответствует $\nabla^2 G$ – фильтр, где ∇^2 – оператор Лапласа, а символ G обозначает распределение Гаусса (5.66) со среднеквадратическим отклонением σ :

$$G(x, y) = \exp\left(-\frac{x^2 + y^2}{2\pi\sigma^2}\right). \quad (5.66)$$

Двумерное распределение значений $\nabla^2 G$ – фильтра можно задать через расстояние $r = \sqrt{x^2 + y^2}$ от начала координат с помощью следующего выражения, называемого также как **лапласиан гауссiana** (или функция вида **мексиканская шляпа** – вторая производная по r от выражения (5.66)):

$$\nabla^2 G(r) = -\frac{1}{\pi\sigma^2} \left(1 - \frac{r^2}{\pi\sigma^2}\right) \exp\left(-\frac{r^2}{2\pi\sigma^2}\right). \quad (5.67)$$

На рис. 5.35 представлен одномерный вариант распределения (5.67). Применение оператора G эквивалентно размытию изображения и применению к нему оператора Лапласа. В ре-

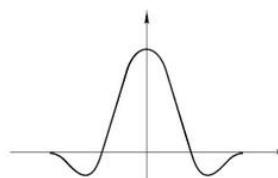


Рис. 5.35. Одномерный вариант лапласиана гауссiana

Числовые характеристики изображения

Локальные признаки: особые точки

Лапласиан гауссиана

В работах [46, 47, 50] показано, что в наибольшей степени требованиям сегментации соответствует $\nabla^2 G$ — фильтр, где ∇^2 — оператор Лапласа, а символ G обозначает распределение Гаусса (5.66) со среднеквадратическим отклонением σ :

$$G(x, y) = \exp\left(-\frac{x^2 + y^2}{2\pi\sigma^2}\right). \quad (5.66)$$

Двумерное распределение значений $\nabla^2 G$ — фильтра можно задать через расстояние $r = \sqrt{x^2 + y^2}$ от начала координат с помощью следующего выражения, называемого также как **лапласиан гауссиана** (или функция вида **мексиканская шляпа** — вторая производная по r от выражения (5.66)):

$$\nabla^2 G(r) = -\frac{1}{\pi\sigma^2} \left(1 - \frac{r^2}{\pi\sigma^2}\right) \exp\left(-\frac{r^2}{2\pi\sigma^2}\right). \quad (5.67)$$

На рис. 5.35 представлен одномерный вариант распределения (5.67). Применение оператора G эквивалентно размытию изображения и применению к нему оператора Лапласа. В ре-

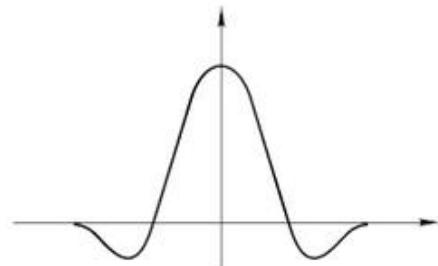


Рис. 5.35. Одномерный вариант лапласиана гауссиана

зультате размытия на изображении уничтожены все структуры, относящиеся к масштабному уровню, существенно меньшему значения пространственной постоянной σ гауссовского распределения. После этого на изображении, полученном в результате фильтрации, отыскиваются точки нулевого уровня, т. е. точки, в которых происходит смена знака.

Иначе говоря, гауссовская функция в лапласиане гауссиана сглаживает изображение (сокращает шум), а непосредственно лапласиан определяет на изображении точки пересечения нулевого уровня, используемые для локализации контуров [46, 47]. Необходимо отметить, что нейрофизиологические эксперименты еще в 1980-х гг. показали, что некоторые аспекты зрения человека описываются математической моделью в форме лапласиана гауссиана (5.67). Если среднеквадратическое отклонение $\sigma = 1$, то маска $\nabla^2 G$ -фильтра эквивалентна данным рис. 5.32.

Фильтрация

Цифровой фильтр — в электронике любой фильтр, обрабатывающий цифровой сигнал с целью выделения и/или подавления определённых частот этого сигнала.

Линейная фильтрация

Линейная оконная фильтрация изображений в пространственной области заключается в вычислении линейной комбинации значений яркости пикселов в окне фильтрации с коэффициентами матрицы весов фильтра, называемой также *маской* или *ядром* линейного фильтра.

Рассмотрим вычисление такой линейной комбинации на примере окна фильтрации размером 3×3 . При этом маска фильтра представляется матрицей вида

$$\begin{matrix} \text{Mask}[-1, -1] & \text{Mask}[0, -1] & \text{Mask}[1, -1] \\ \text{Mask}[-1, 0] & \text{Mask}[0, 0] & \text{Mask}[1, 0] \\ \text{Mask}[-1, 1] & \text{Mask}[0, 1] & \text{Mask}[1, 1], \end{matrix}$$

а соответствующий фрагмент изображения с центральным пикселиом $\text{Im}[x, y]$, к которому на текущем шаге применяется данный фильтр, имеет вид:

$$\begin{matrix} \text{Im}[x-1, y-1] & \text{Im}[x, y-1] & \text{Im}[x+1, y-1] \\ \text{Im}[x-1, y] & \text{Im}[x, y] & \text{Im}[x+1, y] \\ \text{Im}[x-1, y+1] & \text{Im}[x, y+1] & \text{Im}[x+1, y+1]. \end{matrix}$$

Результат линейной фильтрации для данного окна (для данного центрального пикселя) описывается следующей простой формулой:

$$\text{Im}'[x, y] = \sum_{i=-\text{hWinX}}^{\text{hWinX}} \sum_{j=-\text{hWinY}}^{\text{hWinY}} \text{Im}[x+i, y+j] \cdot \text{Mask}[x+i, y+j], \quad (1)$$

где $\text{hWinX} = [\text{WinX}/2]$, $\text{hWinY} = [\text{WinY}/2]$ - полуширина и полувысота окна фильтрации соответственно (в случае окна 3×3 обе величины равны 1).

Результат применения операции (1) ко всем пикселям изображения $\text{Im}[x, y]$ называется сверткой изображения Im с маской Mask .

Скользящее среднее Гауссова маска

$$\frac{1}{9} \times \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}. \quad \frac{1}{16} \times \begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix}.$$

Фильтрация

Нелинейная фильтрация

Нелинейная ранговая фильтрация является непосредственным обобщением бинарной ранговой фильтрации и опирается на понятие порядковой статистики. Вокруг каждого элемента изображения выбирается окрестность, входящие в нее элементы изображения упорядочиваются по возрастанию яркости. Ранговый фильтр порядка r ($1 \leq r \leq N$, где N - число отсчетов в окрестности) выбирает из полученного ряда элемент с номером r и присваивает его значение исходному элементу изображения. Когда число N нечетное и $r=(N+1)/2$, то фильтр называется медианным. Медианный фильтр имеет важное значение в обработке изображений вследствие высокой рабочести, то есть нечувствительности результатов фильтрации к плотности распределения (первого порядка) шумовой компоненты. Это связано с тем, что медианный фильтр с апертурой площадью $2M+1$ эффективно подавляет локальные области площадью менее M пикселов. В то же время, при фильтрации контрастных крупноразмерных объектов медианный фильтр не размывает и не смешает их края (точки перепада яркости).

1	1	1
1	1	1
1	1	1

17	23	27
32	18	35
37	20	21

17, 18, 20, 21, 23, 27, 32, 35, 37

p=1: 17
p=9: 37

Маска

Изображение

Исходное изображение

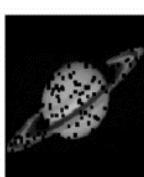


Упорядоченные пиксели

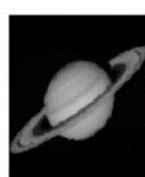
Шум «соль и перец»



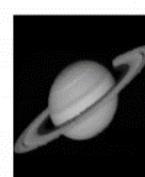
Ранговая фильтрация с маской 3x3



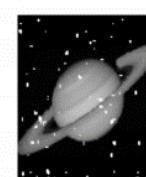
p=1



p=4



p=5



p=9

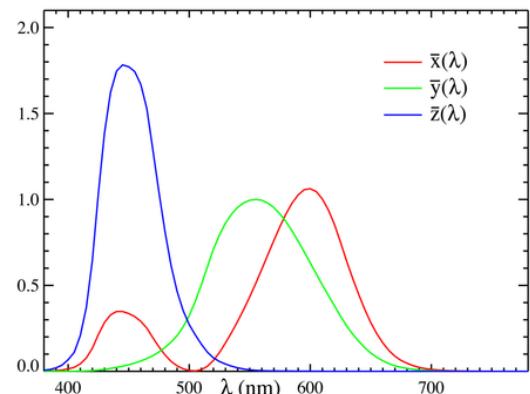
Цветовые модели

CIE XYZ

$$X = \int_{380}^{780} I(\lambda) \bar{x}(\lambda) d\lambda$$

$$Y = \int_{380}^{780} I(\lambda) \bar{y}(\lambda) d\lambda$$

$$Z = \int_{380}^{780} I(\lambda) \bar{z}(\lambda) d\lambda$$



xyY

Если формально построить сечение пространства XYZ плоскостью

$X + Y + Z = const$, то можно две оставшиеся линейно-независимыми координаты записать в виде

$$x = X/(X + Y + Z)$$

$$y = Y/(X + Y + Z).$$

аналогично, но необязательно:

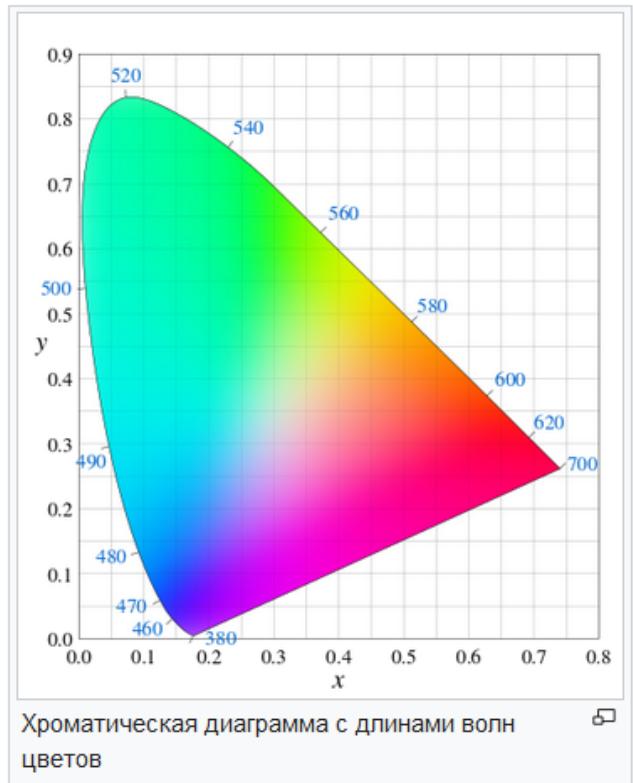
$$z = Z/(X + Y + Z)$$

Такое сечение называется хроматической диаграммой (диаграммой цветности).

В пространстве XYZ точке $(X, 0, 0)$, как легко посчитать по формулам, на хроматической диаграмме соответствует точка $xy=(1, 0)$.

Подобным образом, точке $XYZ=(0, Y, 0)$ соответствует точка $xy=(0, 1)$,

наконец, точке $XYZ=(0, 0, Z)$ — точка $xy=(0, 0)$. Видно, что все реальные цвета, полученные любыми спектральными составами излучений, в том числе и монохроматическими (спектральные цвета) не дотягивают до подобных «чистых» значений. Данная закономерность вытекает из правила смешивания цветов и является проявлением того, что невозможно получить отклик одних колбочек без отклика других (хоть и очень малого), а также из того, что яркость Y не может иметь нулевое или малое значение при определенном отклике любых колбочек.



Хроматическая диаграмма с длинами волн цветов

Цветовые модели

xyY

Для смешения двух цветов используются законы Гассмана. Пусть два цвета заданы на графике МКО координатами $D1=(x_1, y_1, Y_1)$ и $D2=(x_2, y_2, Y_2)$.

Тогда смешение их дает цвет

$$D12=(x_1+x_2, y_1+y_2, z_1+z_2).$$

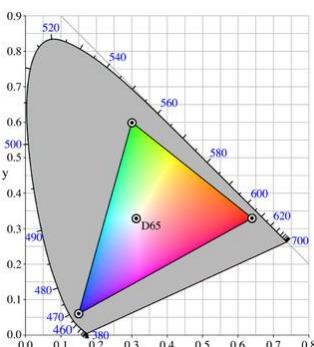
Если ввести обозначения $t1=Y_1/y_1$, $t2=Y_2/y_2$, то получим координаты цветности смеси

$$x12=(x_1 * t1 + x_2 * t2)/(t1+t2),$$

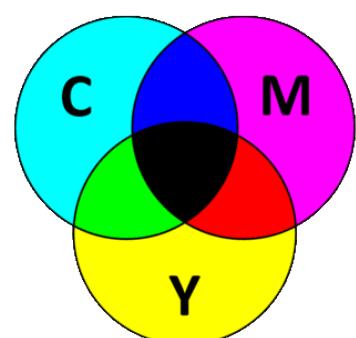
$$y12=(y_1 * t1 + y_2 * t2)/(t1+t2),$$

$$Y12=Y_1+Y_2.$$

RGB



CMYK



Given the chromaticity coordinates of an RGB system (x_r, y_r) , (x_g, y_g) and (x_b, y_b) and its reference white (X_W, Y_W, Z_W) , here is the method to compute the 3×3 matrix for converting RGB to XYZ:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where

$$[M] = \begin{bmatrix} S_r X_r & S_g X_g & S_b X_b \\ S_r Y_r & S_g Y_g & S_b Y_b \\ S_r Z_r & S_g Z_g & S_b Z_b \end{bmatrix}$$

$$X_r = x_r / y_r$$

$$Y_r = 1$$

$$Z_r = (1 - x_r - y_r) / y_r$$

$$X_g = x_g / y_g$$

$$Y_g = 1$$

$$Z_g = (1 - x_g - y_g) / y_g$$

$$X_b = x_b / y_b$$

$$Y_b = 1$$

$$Z_b = (1 - x_b - y_b) / y_b$$

$$\begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix}$$

(2)

Цветовые модели

HSV

HSV (англ. Hue, Saturation, Value — тон, насыщенность, значение) или HSB (англ. Hue, Saturation, Brightness — тон, насыщенность, яркость) — цветовая модель, в которой координатами цвета являются:

- **Hue** — цветовой тон, (например, красный, зелёный или сине-голубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1.
- **Saturation** — насыщенность. Варьируется в пределах 0—100 или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
- **Value** (значение цвета) или Brightness — яркость. Также задаётся в пределах 0—100 или 0—1.

Модель была создана Элви Рейм Смитом, одним из основателей Pixar, в 1978 году. Она является нелинейным преобразованием модели RGB.

RGB → HSV [править | править код]

Считаем, что:

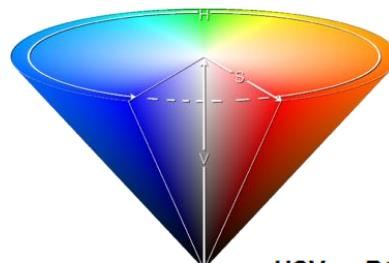
$$H \in [0, 360]$$

$$S, V, R, G, B \in [0, 1]$$

Пусть MAX — максимальное значение из R, G и B , а MIN — минимальное из них.

$$H = \begin{cases} 0, & \text{если } MAX = MIN \\ 60 \times \frac{G - B}{MAX - MIN} + 0, & \text{если } MAX = R \text{ и} \\ & G \geq B \\ 60 \times \frac{G - B}{MAX - MIN} + 360, & \text{если } MAX = R \text{ и} \\ & G < B \\ 60 \times \frac{B - R}{MAX - MIN} + 120, & \text{если } MAX = G \\ 60 \times \frac{R - G}{MAX - MIN} + 240, & \text{если } MAX = B \end{cases}$$
$$S = \begin{cases} 0, & \text{если } MAX = 0; \\ 1 - \frac{MIN}{MAX}, & \text{иначе} \end{cases}$$

$$V = MAX$$



HSV → RGB [править | править код]

Для любых оттенков $H \in [0, 360]$, насыщенности $S \in [0, 100]$ и яркости $V \in [0, 100]$:

$$H_i = \left\lfloor \frac{H}{60} \right\rfloor \mod 6$$
$$V_{min} = \frac{(100 - S) * V}{100}$$
$$a = (V - V_{min}) * \frac{H \mod 60}{60}$$

$$V_{inc} = V_{min} + a$$

$$V_{dec} = V - a$$

H_i	R	G	B
0	V	V_{inc}	V_{min}
1	V_{dec}	V	V_{min}
2	V_{min}	V	V_{inc}
3	V_{min}	V_{dec}	V
4	V_{inc}	V_{min}	V
5	V	V_{min}	V_{dec}



Цветовые модели

HLS

HSL, HLS или HSI (от англ. hue, saturation, lightness (intensity)) — цветовая модель, в которой цветовыми координатами являются тон, насыщенность и светлота. Следует отметить, что HSV и HSL — две разные цветовые модели (lightness — светлота, что отличается от яркости).

RGB to HLS

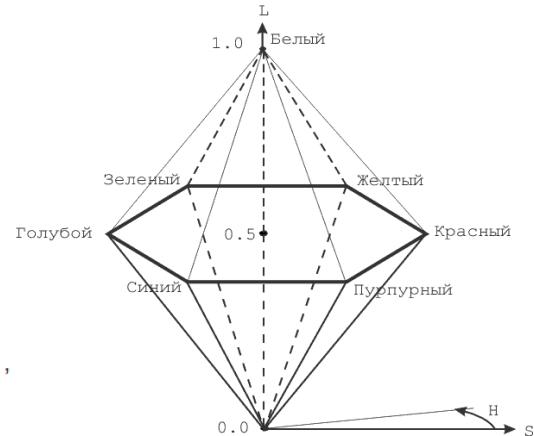
$$H = \begin{cases} \text{undefined} & \text{if } MAX = MIN \\ 60^\circ \times \frac{G-B}{MAX-MIN} + 0^\circ, & \text{if } MAX = R \\ & \text{and } G \geq B \\ 60^\circ \times \frac{G-B}{MAX-MIN} + 360^\circ, & \text{if } MAX = R \\ & \text{and } G < B \\ 60^\circ \times \frac{B-R}{MAX-MIN} + 120^\circ, & \text{if } MAX = G \\ 60^\circ \times \frac{R-G}{MAX-MIN} + 240^\circ, & \text{if } MAX = B \end{cases},$$

$$S = \begin{cases} 0 & \text{if } L = 0 \text{ or } MAX = MIN \\ \frac{MAX-MIN}{MAX+MIN} = \frac{MAX-MIN}{2L}, & \text{if } 0 < L \leq \frac{1}{2} \\ \frac{MAX-MIN}{2-(MAX+MIN)} = \frac{MAX-MIN}{2-2L}, & \text{if } \frac{1}{2} < L < 1 \end{cases},$$

или, в общем случае, $S = \frac{MAX - MIN}{1 - |1 - (MAX + MIN)|}$,

$$L = \frac{1}{2} (MAX + MIN), \text{ где:}$$

- R, G, B — значения цвета в цветовой модели **RGB**, значения в диапазоне [0; 1] (R - красный, G - зелёный, B - синий).
- **MAX** — максимум из трёх значений (R, G, B)
- **MIN** — минимум из трёх значений (R, G, B)
- **H** — тон [0; 360]
- **S** — насыщенность [0; 1]
- **L** — светлота [0; 1]



HLS to RGB

$$Q = \begin{cases} L \times (1.0 + S), & \text{if } L < 0.5 \\ L + S - (L \times S), & \text{if } L \geq 0.5 \end{cases}$$

$$P = 2.0 \times L - Q$$

$$H_k = \frac{H}{360} \text{ (приведение к интервалу [0,1])}$$

$$T_R = H_k + \frac{1}{3}$$

$$T_G = H_k$$

$$T_B = H_k - \frac{1}{3}$$

if $T_c < 0 \rightarrow T_c = T_c + 1.0$ for each $c = R, G, B$

if $T_c > 1 \rightarrow T_c = T_c - 1.0$ for each $c = R, G, B$

Для каждого цвета $c = R, G, B$:

$$\text{color}_c = \begin{cases} P + ((Q - P) \times 6.0 \times T_c), & \text{if } T_c < \frac{1}{6} \\ Q, & \text{if } \frac{1}{6} \leq T_c < \frac{1}{2} \\ P + \left((Q - P) \times \left(\frac{2}{3} - T_c \right) \times 6.0 \right), & \text{if } \frac{1}{2} \leq T_c < \frac{2}{3} \\ P, & \text{otherwise} \end{cases}$$

Цветовые модели

Монохромное изображение

Существует два типа растровых изображений, которые можно относить к монохромным:

Бинарное изображение

Бинарное (двоичное) изображение иногда могут называть «монохромным» и «чёрно-белым», что в общем случае не верно и ведёт иногда к путанице.

Бинарное растровое изображение может являться монохромным в случаях, когда:

- один вид пикселей (не важно «0» или «1») отображается абсолютно чёрным цветом, а другой любым произвольным (например, «чёрно-белый» растр, «чёрно-зелёный» и т. д.);
- оба вида (и «0», и «1») отображаются одним оттенком, но с разной яркостью (например, «тёмно- и светло- серый», «ярко-зелёный и тёмно-зелёный», «синий и серый» и т. д.).

Бинарное изображение не будет являться монохромным, если разные виды пикселей отображаются разными оттенками цвета (например, «жёлто-зелёное», «красно-синее» растровое изображение и пр.).

Полутоновое изображение

Полутоновое растровое изображение всегда является монохромным по определению, независимо от того полутона (яркости) какого оттенка цвета оно содержит.

Grayscale

В цветовых пространствах [YUV](#) и [YIQ](#) используемые в [PAL](#) и [NTSC](#) яркость Y' вычисляется следующим образом:

$$Y' = 0.299R + 0.587G + 0.114B$$

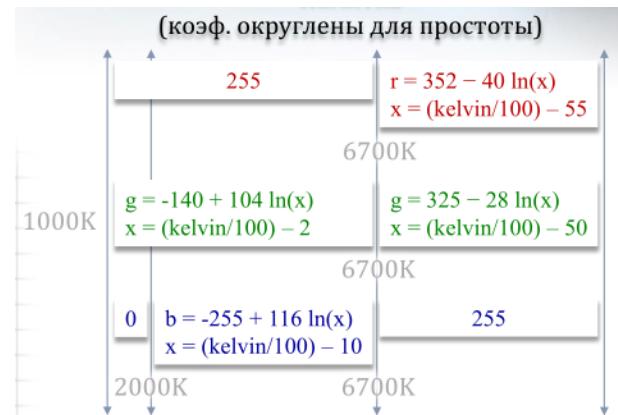
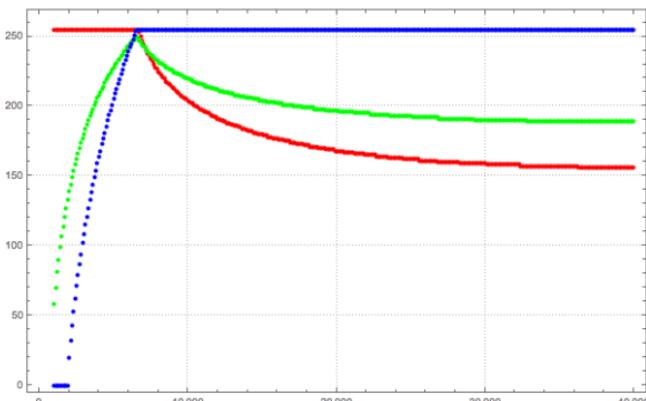
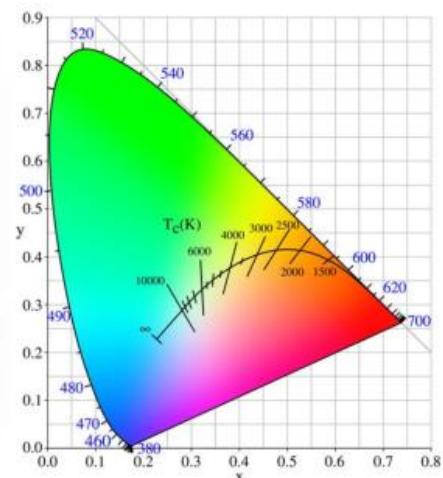
Для учёта особенностей восприятия изображения человеческим глазом (чувствительность к зелёному и синему цвету) в модели [HDTV](#) используют другие коэффициенты:

$$Y' = 0.2126R + 0.7152G + 0.0722B$$

Температура изображения

Абсолютно чёрное тело — физическое тело, которое при любой температуре поглощает всё падающее на него электромагнитное излучение во всех диапазонах

Цветовая температура источника света — это температура абсолютно чёрного тела, излучающего свет, сравнимый с цветом источника света. Измеряется в кельвинах или майродах.



Алгоритм коррекции цвета

1
Подсчет цвета
температуры в RGB

$$K \rightarrow r \ g \ b$$

2
Для каждого пикселя:

2.1
Посчитать яркость
пикселя (точнее,
«светлоту»)

$$R \ G \ B$$

$$R \ G \ B \rightarrow \text{luminance}$$

$$\text{Lum} = (\min(R, G, B) + \max(R, G, B)) / 2$$

2.2
Скорректировать цвет
пикселя изображения с
учетом температурных
добавок

$$R \ G \ B, \ r \ g \ b, \ alpha \rightarrow$$

$$R = (1 - \alpha)*R + \alpha*r$$

$$G = (1 - \alpha)*G + \alpha*g$$

$$B = (1 - \alpha)*B + \alpha*b$$

If > 255, then = 255

2.3
Перевести новые RGB в
HLS

$$R \ G \ B \rightarrow H \ L \ S$$

2.4
Заменить компоненту L
подсчитанной ранее
яркостью пикселя

$$L = \text{Lum}$$

2.5
Перевести обратно из
HLS в RGB

$$H \ L \ S \rightarrow R \ G \ B$$

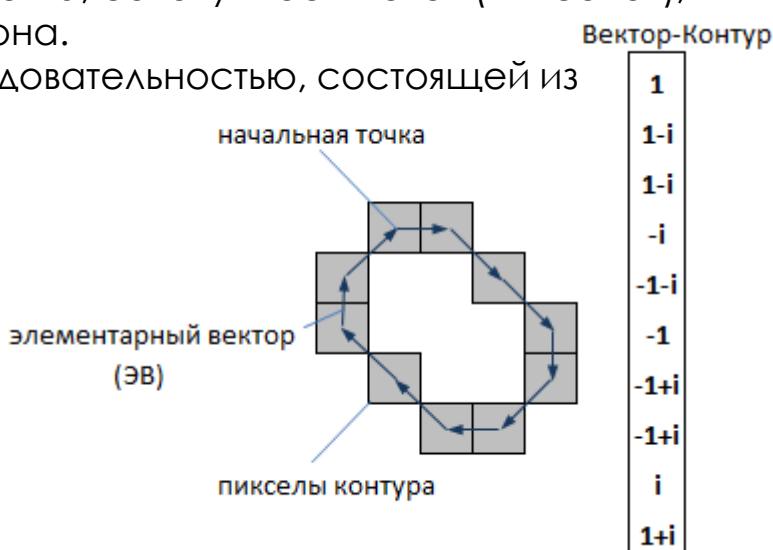
Контурный анализ

- позволяет описывать, хранить, сравнивать и производить поиск объектов (по контурам)
- рассмотрение только контуров позволяет снизить вычислительную и алгоритмическую сложность
- методы контурного анализа инвариантны к некоторым преобразованиям (перенос, поворот и изменение масштаба)

Вектор-контур

- Контур – это граница объекта, совокупность точек (пикселов), отделяющих объект от фона.
- Контур кодируется последовательностью, состоящей из комплексных чисел.
- Начальная точка.
- Вектор смещения
- Элементарный вектор
- Вектор-контур
- Вектор-контур Γ длины k будем обозначать:

$$\Gamma = (\gamma_0, \gamma_1, \dots, \gamma_{k-1})$$



Свойства контуров

- Сумма ЭВ замкнутого контура равна нулю.
- Контур-вектор не зависит от параллельного переноса исходного изображения.
- Поворот изображения на определенный угол равносителен повороту каждого ЭВ контура на тот же угол.
- Изменение начальной точки ведет к циклическому сдвигу ВК.
- Изменение масштаба исходного изображения можно рассматривать как умножение каждого ЭВ контура на масштабный коэффициент.

Скалярное произведение векторов

$$\eta = (\Gamma, N) = \sum_{n=0}^{k-1} (\gamma_n, v_n)$$

(γ_n, v_n) — скалярное произведение комплексных чисел, вычисляемое как: (только для одинаковых размерностей)
 $(a + ib, c + id) = (a + ib)(c - id) = ac + bd + i(bc - ad)$

Контурный анализ

Нормированное скалярное произведение (НСП)

$$\eta = \frac{(\Gamma, N)}{|\Gamma||N|}$$

где $|\Gamma|$ и $|N|$ — нормы(длины) контуров, вычисляемые как:

$$|\Gamma| = \left(\sum_{n=0}^{k-1} |\gamma_n|^2 \right)^{\frac{1}{2}}$$

- НСП в пространстве комплексных чисел, также является комплексным числом.
- Модуль НСП достигает максимального значение – единицы, только если контур Γ является тем же контуром N , но повернутым на некоторый угол и промасштабированный на определенный коэффициент.
- Модуль нормированного скалярного произведения контуров даст единицу только в том случае, если эти два контура равны с точностью до поворота и масштаба.
- В противном случае, модуль НСП будет строго меньше единицы.
- Модуль дает меру сходства контуров
- Аргумент НСП – угол поворота контуров относительно друг друга.

Недостаток НСП

- Выбор начальной точки
- Если контуры одинаковы, но отсчет ЭВ начинается с другой начальной точки, то модуль НСП таких контуров не будет равен единице.

Взаимокорреляционная функция (ВКФ)

$$\tau(m) = (\Gamma, N^{(m)}), \quad m = 0, \dots, k - 1$$

Где $N(m)$ — контур, полученный из N путем циклического сдвига его ЭВ на m элементов.

- Значения ВКФ показывают насколько похожи контуры Γ и N , если сдвинуть начальную точку N на m позиций.
- ВКФ определена на всем множестве целых чисел.
- ВКФ является периодической, с периодом k .

Контурный анализ

Взаимокорреляционная функция

$$\tau_{max} = \max \left(\frac{\tau(m)}{|\Gamma||N|} \right), \quad m = 0, \dots, k - 1$$

- τ_{max} является мерой похожести двух контуров, инвариантной переносу, масштабированию, вращению и сдвигу начальной точки.
- $\arg(\tau_{max})$ дает угол поворота одного контура, относительно другого.

Автокорреляционная функция (АКФ)

- Это скалярное произведение контура самого на себя при различных сдвигах начальной точки:
 $v(m) = (\Gamma, \Gamma^{(m)})$, $m = 0, \dots, k - 1$

Свойства АКФ

- АКФ не зависит от выбора начальной точки контура
- Модуль АКФ симметричен относительно центрального отсчета $k/2$.
- Если контур имеет какую-либо симметрию относительно поворота, то аналогичную симметрию имеет его АКФ.
- АКФ контура можно считать характеристикой формы контура.
- Нормированная АКФ не зависит от масштаба, положения, вращения и выбора начальной точки контура.

Задача распознавания

Последовательность действия при распознавании выглядит так:

- Предварительная обработка изображения — сглаживание, фильтрация помех, повышение контраста.
- Бинаризация изображения и выделение контуров объектов.
- Начальная фильтрация контуров по периметру, площади, коэффициенту формы, фрактальности и так далее.
- Приведение контуров к единой длине, сглаживание.
- Перебор всех найденных контуров, поиск шаблона, максимально похожего на данный контур.

Контурный анализ

Дескриптор контура

- Дескриптор – величина, характеризующая форму контура. При этом, близкие между собой контуры должны иметь близкие дескрипторы.
- АКФ инвариантно к переносу, вращению, масштабированию и выбору начальной точки. И кроме того,
- АКФ является функцией одного контура
- АКФ можно выбрать в качестве дескриптора, описывающего форму контура

Свертка АКФ

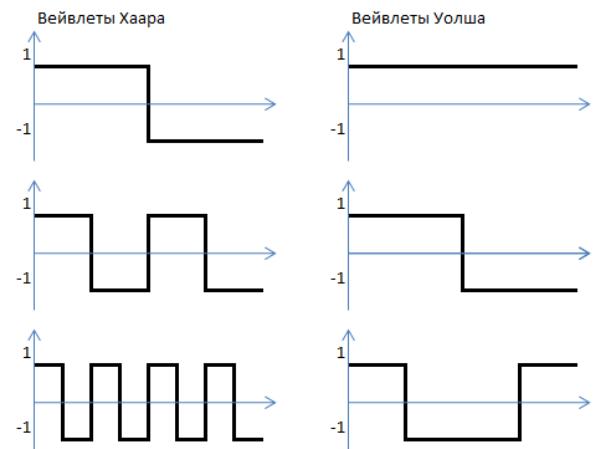
- АКФ - вектор с $k/2$ значениями
- Вейвлетная свертка позволит нам упорядочить значения АКФ в масштабном порядке. Первым будет идти компонент, отвечающий наиболее крупномасштабным особенностям АКФ, а дальнейшие компоненты будут уточнять все более мелкие особенности АКФ.

Особенности сравнения АКФ:

- Сравнение АКФ, в общем случае, не избавляет нас от необходимости вычисления ВКФ.
- Иногда сравнения АКФ может быть достаточно для идентификации контуров.
- Первый компонент свертки АКФ дает нам хороший дескриптор для упорядочивания базы шаблонов.

Недостатки КА

- Проблема выделения контура на изображениях.
- Описывает весь объект целиком, и не допускает никаких пересечений с другими объектами или неполной видимости объекта.



Заключение

- Методы КА привлекательны своей простотой и быстродействием.
- При наличии четко выраженного объекта на контрастном фоне и отсутствии помех КА хорошо справляется с распознаванием.

Методы выделения контуров

Основанные на вычислении градиента

Большинство методов, которые применяются для выделения контуров изображений основаны на вычислении градиента изображения.

Наиболее известными операторами этого класса методов являются операторы Собеля, Прюитта и Робертса.

Результат линейной фильтрации для данного окна (для данного центрального пикселя) описывается следующей простой формулой:

$$Im'[x, y] = \sum_{i=-hWinX}^{hWinX} \sum_{j=-hWinY}^{hWinY} Im[x + i, y + j] \cdot Mask[x + i, y + j],$$

где $hWinX = [WinX/2]$, $hWinY = [WinY/2]$ - полуширина и полувысота окна фильтрации соответственно (в случае окна 3×3 обе величины равны 1).

Вычисление производных

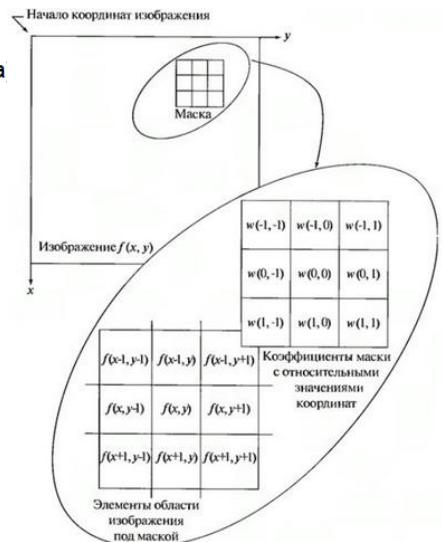
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad \text{- градиент изображения } \mathbf{f}(x,y) \text{ в точке } (x,y) \quad (4)$$

$$|\nabla f| = \sqrt{G_x^2 + G_y^2} \quad \text{- модуль градиента изображения } \mathbf{f}(x,y) \text{ в точке } (x,y) \quad (5)$$

$$\alpha(x,y) = \arctg\left(\frac{G_y}{G_x}\right) \quad \text{- угол между направлением вектора } \nabla f \text{ в точке } (x,y) \text{ и осью } x$$



Отсюда легко найти направление контура в точке (x,y) , которое перпендикулярно направлению вектора градиента в этой точке. А вычислить градиент изображения можно, вычислив величины частных производных $\partial f / \partial x$ и $\partial f / \partial y$ для каждой точки.

Перекрестный оператор Робертса

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

- окрестность 3×3 внутри изображения

$$G_x = (z_9 - z_5) \quad (7)$$

$$G_y = (z_8 - z_6) \quad (8)$$

-1	0
0	1

- маски оператора Робертса

Методы выделения контуров

Основанные на вычислении градиента

Оператор Превитта

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

- окрестность 3x3 внутри изображения

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \quad (9)$$

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \quad (10)$$

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

- маски оператора
Превитта

Оператор Собеля

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

- окрестность 3x3 внутри изображения

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (1)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (1)$$

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

- маски оператора Собеля

Фильтр Канни(детектор)

Сглаживание. Размытие изображения для удаления шума.

Поиск градиентов. Границы отмечаются там, где градиент изображения приобретает максимальное значение. Они могут иметь различное направление, поэтому алгоритм Кэнни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в размытом изображении.

Угол направления вектора градиента округляется и может принимать такие значения: 0, 45, 90, 135.

Подавление немаксимумов. Только локальные максимумы отмечаются как границы.

Двойная пороговая фильтрация. Потенциальные границы определяются порогами.

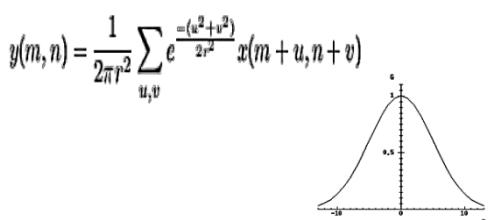
Трассировка области неоднозначности. Итоговые границы определяются путём подавления всех краёв, несвязанных с определенными (сильными) границами.

Методы выделения контуров

Основанные на поиске нулей

Методы, основанные на поиске нулей, ищут пересечения оси абсцисс выражения второй производной, обычно нули лапласиана или нули нелинейного дифференциального выражения, как будет описано далее. В качестве шага предобработки к выделению границ практически всегда применяется сглаживание изображения, обычно фильтром Гаусса.

Фильтр Гаусса



Лаплассиан

-1	-1	-1
-1	8	-1
-1	-1	-1

$$\partial_v(L_v) = 0$$

$$\partial_{vv}(L_v) \leq 0$$

$$L_v^2 L_{vv} = L_x^2 L_{xx} + 2 L_x L_y L_{xy} + L_y^2 L_{yy} = 0$$

$$L_v^3 L_{vvv} = L_x^3 L_{xxx} + 3 L_x^2 L_y L_{xxy} + 3 L_x L_y^2 L_{xyy} + L_y^3 L_{yyy} \leq 0$$

где $L_x, L_y \dots L_{yyy}$ - частные производные, посчитанные на масштабном представлении L , полученном с помощью фильтрации исходного изображения фильтром Гаусса

$$L_{xx}(x, y) = L(x-1, y) - 2L(x, y) + L(x+1, y).$$

$$L_{xy}(x, y) = (L(x-1, y-1) - L(x-1, y+1) - L(x+1, y-1) + L(x+1, y+1))/4,$$

$$L_{yy}(x, y) = L(x, y-1) - 2L(x, y) + L(x, y+1).$$

$$L_{xx} = [1 \ -2 \ 1] * L \quad \text{and} \quad L_{xy} = \begin{bmatrix} -1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4 \end{bmatrix} * L \quad \text{and} \quad L_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} * L$$

Алгоритмы сегментации изображения

Постановка задачи

- На изображении присутствует область (объект), отличающаяся от остальной части изображения (фона).
- Задача: выделить объект на изображении, отделив его от фона.

Волшебная палочка

- Составить матрицу размером $n \times m$, элементы которой заданы формулой:
- $$B(x, y) = \begin{cases} 1, & \text{если } \rho(c_{xy}, c_0) \leq T \\ 0, & \text{иначе} \end{cases}$$
 где c_{xy} – цвет пикселя с индексами x и y , T – заданный порог чувствительности.
- Данная матрица показывает, какие пиксели отличаются от заданного не более, чем на порог T .
- Чем больше T , тем большую область алгоритм отметит, как объект.
- При слишком больших значениях T часть фона может быть отмечена, как объект; при слишком малых – часть объекта может быть воспринята, как фон.

Обозначения

- Размер изображения – $n \times m$ пикселей;
- $\rho(c_1, c_2)$ – функция, определяющая расстояние между цветами c_1 и c_2 ;
- c_0 – цвет выбранного пользователем пикселя.

Волшебная палочка

- Для полученной матрицы B с помощью любого алгоритма поиска связных областей найти область, которой принадлежит выбранный пользователем пиксель. Принадлежащие этой области пиксели и будут результатом работы алгоритма.
- Алгоритм плохо работает с пёстрыми объектами и размытыми границами.

«Волшебная палочка»



«Умные ножницы»

- Пользователь последовательно указывает несколько точек, принадлежащих границе объекта
- Алгоритм самостоятельно достраивает границу
- Представим изображение как граф, вершины которого – углы пикселей, а рёбра – стороны пикселей.
- Длина ребра определяется по формуле:

$$d = \frac{L}{K + \rho(c_1, c_2)}$$

где L – геометрическая длина ребра, c_1 и c_2 – цвета пикселей по обе стороны ребра, $K = \text{const}$

- С учётом полученных длин рёбер между заданными пользователем точками находится кратчайший путь, который при замыкании образует границу между объектом и фоном.

- Алгоритм плохо работает с пёстрыми изображениями. В этом случае пользователю придётся указывать большое число граничных точек.

«Умные ножницы»



Алгоритмы сегментации изображения

Разрезы на графах

- Наиболее удобный для пользователя интерфейс: пользователь указывает часть пикселей, принадлежащих объекту, и часть пикселей, принадлежащих фону, после чего алгоритм производит сегментацию.
- Если результат неудовлетворителен, пользователь отмечает новые пиксели как фон/объект, добавляя их к отмеченным ранее, после чего результат пересчитывается.
- Представим изображение в виде графа. Вершинами принимаются центры пикселей, затем для каждого пикселя соответствующая вершина соединяется ребрами с вершинами, которые соответствуют соседним пикселям.
- Вес ребра определяется по формуле:

$$d = \lambda \frac{1}{L} e^{-\sigma \rho(c_1, c_2)}$$

где L – геометрическая длина ребра, c_1 и c_2 – цвета соединяемых ребром вершин, λ и σ – положительные константы.

Чем меньше вес, тем больше разница между цветами соединяемых вершин.

- Пусть пользователь отметил два множества пикселей: множество A содержит пиксели, относящиеся к объекту, а множество B – пиксели, относящиеся к фону.
- Добавим две новые вершины – *сток* и *исток*. Эти вершины соединяются ребрами со всеми остальными вершинами, причём вес ребер между истоком и вершинами множества A равен бесконечности, как и вес ребер между стоком и вершинами множества B.
- Для вершин, не принадлежащих множествам A и B, вес ребер, соединяющих их с истоком и стоком соответственно, задаётся пропорционально схожести их цвета и цветов всех остальных вершин множества: чем сильней цвет вершины похож на цвета остальных вершин множества A (B), тем больше вес ребра, соединяющего её с истоком (стоком).

- Разрез* – разбиение вершин графа на два множества.
- Рёбра, соединяющие вершины из разных множеств, называются *разрезанными*.
- Вес разреза* – сумма весов всех разрезанных рёбер.
- Минимальный разрез* – разрез с минимальным весом.
- Найдём минимальный разрез (например, с помощью алгоритма поиска максимального потока в сети).
- Полученный разрез разделит все вершины на принадлежащие объекту или фону. При этом вершины из A будут отнесены к объекту, вершины из B – к фону.
- Граница между объектом и фоном будет проведена по возможности между пикселями с сильно отличающимися цветами.
- Пиксели, похожие по цвету на пиксели множества A, будут по возможности отнесены к объекту, а пиксели, похожие по цвету на пиксели множества B, - к фону.



Перевод изображения в монохромное

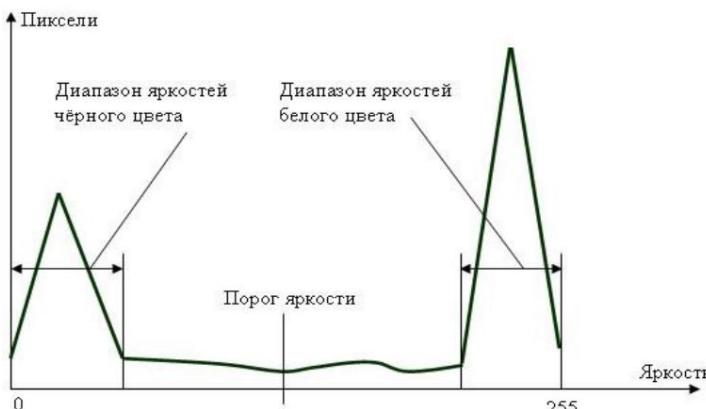
- Имеется массив данных **bytes**, в котором хранится 24-битное изображение в несжатом формате.
- Известны его размеры **Width** и **Height**.
- Сигнал яркости **Y** формируется из RGB сигнала по следующей формуле:

$$Y = 0.222R + 0.707G + 0.071B$$

$$Y' = 0.2126R + 0.7152G + 0.0722B$$

- Задав изначально пороговый уровень **border_brightness** изображение переводится в монохромное путём сравнения больше - меньше этого уровня.
- Необходимо задать пороговый уровень яркости изображения. Например, при распознавании штрих-кодов можно построить гистограмму распределения значений яркости, по которой определить необходимый пороговый уровень.

Бинарный подход



Адаптивное преобразование

- Если изображение освещено неравномерно, бинарный алгоритм неэффективен.
- Необходимо использовать адаптивное преобразование.

Суть метода такова:

- для каждого пикселя определяются окрестности, в которых вычисляется среднее значение яркости, которое и является пороговым значением **T(x,y)** для текущего пикселя.

$$dst(x,y) = \begin{cases} RGB(255,255,255), & \text{если Яркость}(src(x,y)) > T(x,y) \\ RGB(0,0,0), & \text{в другом случае} \end{cases}$$

где **src(x,y)** – RGB-значения пикселя с координатами **x** и **y** на исходном изображении, **dst(x,y)** – соответственно конечного изображения.

Исходное изображение



Бинарное преобразование



Адаптивное преобразование



Нормализация изображения

Нормализация - преобразование изображения, позволяющее привести его к виду, удобному для распознавания.

Последнее подразумевает некий стандарт для нормализованного изображения, в качестве которого могут использоваться средняя яркость, разброс или дисперсия яркости на изображении, ориентация изображенного объекта, его размеры и т. д.

Направления распознавания объектов

- Распознавание или классификация самих изображений;
- Поиск и распознавание объектов (специфических локальных областей) на изображении.

Нормализация изображения в целом называется глобальной нормализацией, нормализация фрагментов изображения – локальной.

Виды

- Яркостная нормализация,
- Нормализация масштаба объекта,
- Нормализация положения объекта,
- Нормализация ориентации объекта.

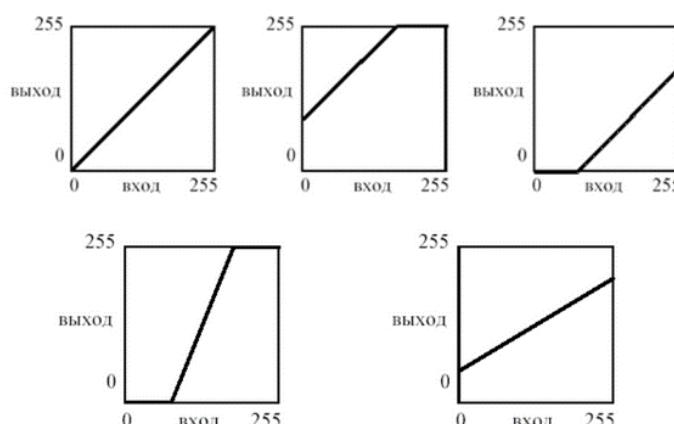


Яркостная и контрастная нормализация

Яркость - степень отличия цвета пикселей от черного цвета (мат. ожидание)

Контраст - разброс цветов пикселей изображения (дисперсия)

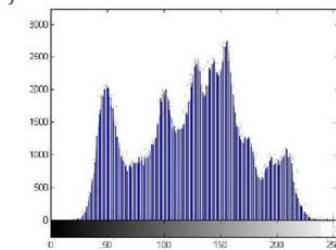
Изменение яркости и контраста



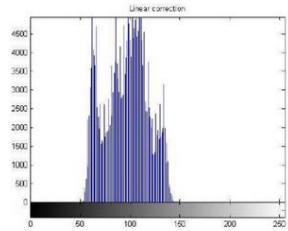
Нормализация изображения

Гистограмма - график распределения яркости на изображении. По оси абсциссы отложены яркости, по оси ординат - кол-во пикселей.

$$h_f[k] = \sum_i \sum_j \begin{cases} 1, & f[i,j] = k \\ 0, & \text{иначе} \end{cases}$$

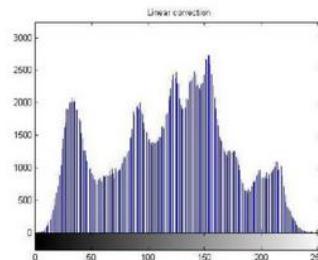


Слабая чувствительность



Линейная коррекция гистограммы

$$y = \frac{y_{max} - y_{min}}{x_{max} - x_{min}}(x - x_{min}) + y_{min}$$



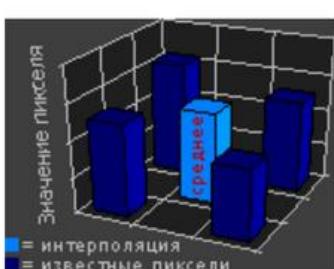
Масштабирование изображений

Метод ближайшего соседа

Для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования.

Билинейная интерполяция

Билинейная интерполяция рассматривает квадрат 2×2 известных пикселей, окружающих неизвестный. В качестве интерполированного значения используется взвешенное усреднение этих четырёх пикселей. В результате изображения выглядят значительно более гладко, чем результат работы метода ближайшего соседа.



1	3	6	1
3	3	1	7
8	1	8	7
6	1	1	1

Набор растровых данных

1	3	6	1
3	3	1	7
8	1	8	7
6	1	1	1

Пересчет по методу Ближайшего соседа

3	3	4	4
4	3	6	6
4	3	4	4
4	2	3	4

Билинейная интерполяция

Нормализация изображения

Поворот изображения

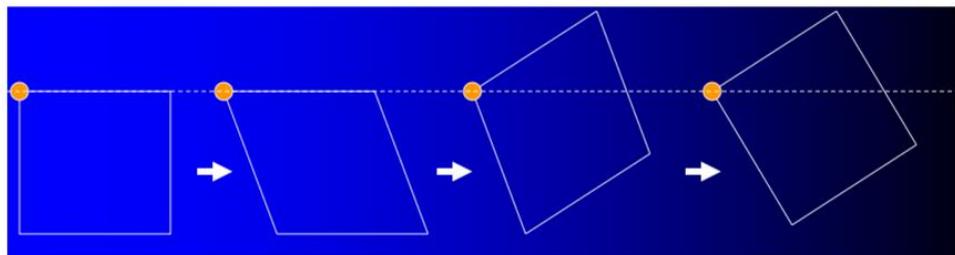
Алгоритм Оуена и Македона

Суть алгоритма Оуэна и Македона заключается в повороте цифровых изображений путём трёх последовательных сдвигов строк, столбцов и затем снова строк изображения.

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \boxed{\begin{bmatrix} \varphi \\ \omega \end{bmatrix}}$$



$$R(\alpha) = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} = \begin{bmatrix} 1 & -\tan \alpha / 2 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ \sin \alpha & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -\tan \alpha / 2 \\ 0 & 1 \end{bmatrix}.$$



Кластеризация

Понятие кластеризации

Кластеризация (или кластерный анализ) — это задача разбиения множества объектов на группы, называемые кластерами.

- Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных групп должны быть как можно более отличны.
- Главное отличие кластеризации от классификации состоит в том, что перечень групп четко не задан и определяется в процессе работы алгоритма.

Этапы кластеризации

- Отбор выборки объектов для кластеризации.
- Определение множества переменных, по которым будут оцениваться объекты в выборке. При необходимости – нормализация значений переменных.
- Вычисление значений меры сходства между объектами.
- Применение метода кластерного анализа для создания групп сходных объектов (кластеров).
- Представление результатов анализа

Меры расстояний

- составить вектор характеристик для каждого объекта
- можно провести нормализацию, чтобы все компоненты давали одинаковый вклад при расчете «расстояния».
- для каждой пары объектов измеряется «расстояние» между ними — степень похожести.

Евклидово расстояние

$$\rho(x, x') = \sqrt{\sum_i^n (x_i - x'_i)^2}$$

Квадрат евклидова расстояния

$$\rho(x, x') = \sum_i^n (x_i - x'_i)^2$$

Расстояние городских кварталов (манхэттенское расстояние) – среднее разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается .

$$\rho(x, x') = \sum_i^n |x_i - x'_i|$$

Кластеризация

Расстояние Чебышева. Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате.

$$\rho(x, x') = \max(|x_i - x'_i|)$$

Степенное расстояние. Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются

$$\rho(x, x') = \sqrt[r]{\sum_i^n (x_i - x'_i)^p}$$

где r и p – параметры, определяемые пользователем. Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами.

Алгоритмы кластеризации

Алгоритмы иерархической кластеризации

Восходящие и нисходящие алгоритмы.

Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры.

Более распространены **восходящие** алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. Таким образом строится система вложенных разбиений.

Алгоритмы квадратичной ошибки

Задачу кластеризации можно рассматривать как построение оптимального разбиения объектов на группы. При этом оптимальность может быть определена как требование минимизации среднеквадратической ошибки разбиения.

$$e^2(X, L) = \sum_{j=1}^K \sum_{i=1}^{n_j} \|x_i^{(j)} - c_j\|^2$$

Кластеризация

Алгоритмы кластеризации

Нечеткие алгоритмы

Наиболее популярным алгоритмом нечеткой кластеризации является алгоритм с-средних (c-means). Он представляет собой модификацию метода k-средних.

Алгоритмы, основанные на теории графов

Суть таких алгоритмов заключается в том, что выборка объектов представляется в виде графа $G=(V, E)$, вершинам которого соответствуют объекты, а ребра имеют вес, равный «расстоянию» между объектами.

Алгоритм выделения связных компонент

В алгоритме выделения связных компонент задается входной параметр R и в графе удаляются все ребра, для которых «расстояния» больше R . Соединенными остаются только наиболее близкие пары объектов. Смысл алгоритма заключается в том, чтобы подобрать такое значение R , лежащее в диапазон всех «расстояний», при котором граф «развалится» на несколько связных компонент. Полученные компоненты и есть кластеры.

Алгоритм выделения связных компонент

Алгоритм минимального покрывающего дерева сначала строит на графе минимальное покрывающее дерево, а затем последовательно удаляет ребра с наибольшим весом.

Алгоритм выделения связных компонент

Алгоритм по слойной кластеризации основан на выделении связных компонент графа на некотором уровне расстояний между объектами (вершинами). Уровень расстояния задается порогом расстояния c .

Кластеризация

Алгоритм k-means

- Число кластеров k задано заранее
- Объекты представляются точками в n -мерном векторном пространстве $X = \{x^{(1)}, \dots, x^{(m)}\} ; x^{(i)} \in \mathbb{R}^n$
- Каждая точка попадает в один из k кластеров

Выбор k

- Предварительное знание о количестве кластеров
- Как много кластеров требуется для текущего приложения
- Типы кластеров, найденные путем экспериментирования с различными значениями k
- Каждый из k кластеров представлен одной точкой (центроидой) в C
- Метрика в пространстве признаков:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}$$

- Цель: минимизировать функцию стоимости

$$\text{Cost} = \sum_{i=1}^m (\arg \min_j \|x_i - c_j\|_2^2)$$

Алгоритм

- Инициализация центроидов случайным образом
- Распределение объектов по кластерам в соответствии с формулой:

$$\text{Cost} = \sum_{i=1}^m (\arg \min_j \|x_i - c_j\|_2^2)$$

- Вычисление центра для каждого кластера как среднее всех точек кластера:
- Если центроиды изменились, переход к 2.

$$c_k := \frac{1}{s_k} \sum X_k$$

Проблемы

- Выбор оптимального значения k
- «Пустые» кластеры
- На каждой итерации $m * k$ сравнений

Кластеризация

Алгоритм k-means

Применение

- Кластеризация
- Создание фильтров в алгоритмах глубокого обучения
- Сегментация изображения



FUZZY c-means

Алгоритмом нечеткой кластеризации - с-средних (c-means).
Модификация метода k-средних.

Шаги алгоритма

- Выбрать начальное нечеткое разбиение n объектов на k кластеров путем выбора матрицы принадлежности U размера $n \times k$.
- Используя матрицу U , найти значение критерия нечеткой ошибки:

$$E^2(X, U) = \sum_{i=1}^N \sum_{k=1}^K U_{ik} \|x_i^{(k)} - c_k\|^2$$

где c_k — «центр масс» нечеткого кластера k : $c_k = \sum_{i=1}^N U_{ik} x_i$

- Перегруппировать объекты с целью уменьшения этого значения критерия нечеткой ошибки.
- Возвращаться в п. 2 до тех пор, пока изменения матрицы U не станут незначительными.

Алгоритм подходит, если:

- заранее неизвестно число кластеров,
- необходимо однозначно отнести каждый объект к одному кластеру.

Кластеризация

Иерархические алгоритмы

Задача кластеризации

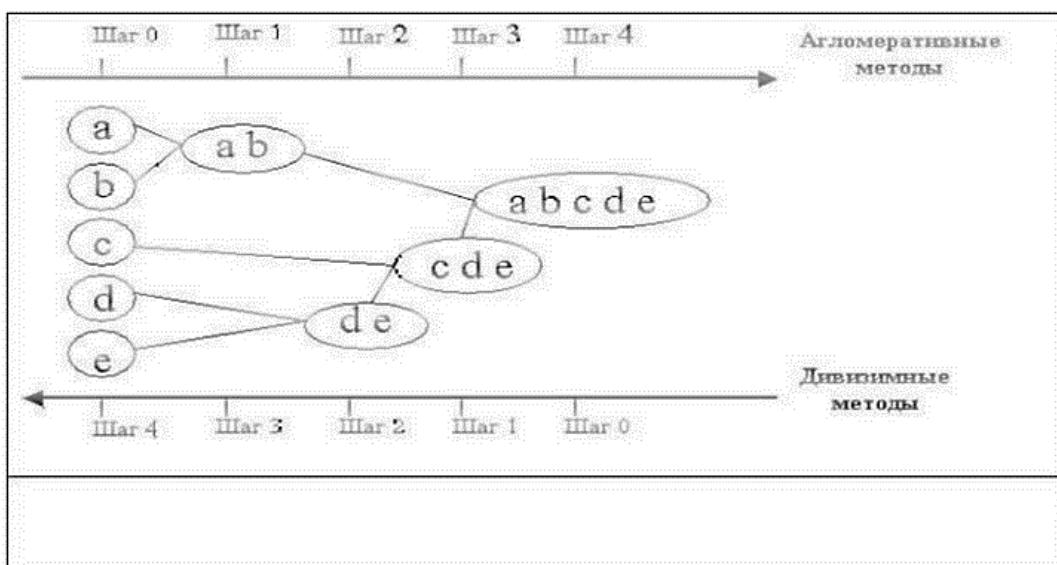
Задача кластеризации (или обучения без учителя) заключается в следующем. Имеется обучающая выборка $X^\ell = \{x_1, \dots, x_\ell\} \subset X$ и функция расстояния между объектами $\rho(x, x')$. Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ , а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^\ell$ приписывается метка (номер) кластера y_i .

Алгоритм кластеризации – это функция $a: X \rightarrow Y$, которая любому объекту $x \in X$ ставит в соответствие метку кластера $y \in Y$. Множество меток Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Особенности иерархической кластеризации (таксономии)

- Построение системы вложенных разбиений
- Результат в виде дерева-дендограммы
- Два типа алгоритмов: восходящие (агломеративные) и нисходящие (дивизимные)
- Сильная зависимость результата от выбранного способа задать расстояние между кластерами

Дендограмма



Кластеризация

Иерархические алгоритмы

Алгоритм Ланса-Уильямса

- 1: инициализировать множество кластеров C_1 :
 $t := 1; C_t = \{\{x_1\}, \dots, \{x_\ell\}\};$
- 2: для всех $t = 2, \dots, \ell$ (t — номер итерации):
 - 3: найти в C_{t-1} два ближайших кластера:
 $(U, V) := \arg \min_{U \neq V} R(U, V);$
 $R_t := R(U, V);$
 - 4: изъять кластеры U и V , добавить слитый кластер $W = U \cup V$:
 $C_t := C_{t-1} \cup \{W\} \setminus \{U, V\};$
 - 5: для всех $S \in C_t$
 - 6: вычислить расстояние $R(W, S)$ по формуле Ланса-Уильямса;

Понятие расстояния

Как определить расстояние $R(W, S)$ между кластерами $W = U \cup V$ и S , зная расстояния $R(U, S), R(V, S), R(U, V)$?

Формула, обобщающая большинство разумных способов определить это расстояние [Ланс, Уильямс, 1967]:

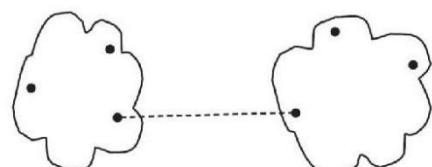
$$\begin{aligned} R(U \cup V, S) = & \alpha_U \cdot R(U, S) + \\ & + \alpha_V \cdot R(V, S) + \\ & + \beta \cdot R(U, V) + \\ & + \gamma \cdot |R(U, S) - R(V, S)|, \end{aligned}$$

где $\alpha_U, \alpha_V, \beta, \gamma$ — числовые параметры.

Способы задания расстояния $R(W, S)$

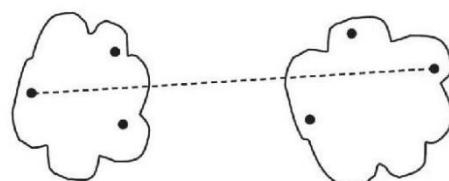
1. Расстояние ближнего соседа:

$$R^b(W, S) = \min_{w \in W, s \in S} \rho(w, s);$$
$$\alpha_U = \alpha_V = \frac{1}{2}, \quad \beta = 0, \quad \gamma = -\frac{1}{2}.$$



2. Расстояние дальнего соседа:

$$R^d(W, S) = \max_{w \in W, s \in S} \rho(w, s);$$
$$\alpha_U = \alpha_V = \frac{1}{2}, \quad \beta = 0, \quad \gamma = \frac{1}{2}.$$



Кластеризация

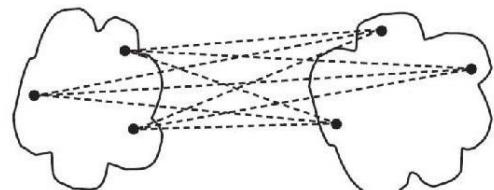
Иерархические алгоритмы

Способы задания расстояния $R(W, S)$

3. Групповое среднее расстояние:

$$R^g(W, S) = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} \rho(w, s);$$

$$\alpha_U = \frac{|U|}{|W|}, \quad \alpha_V = \frac{|V|}{|W|}, \quad \beta = \gamma = 0.$$

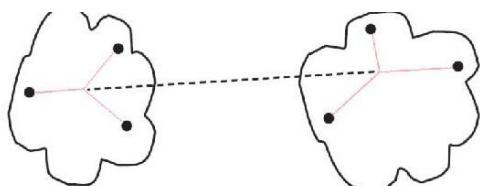


4. Расстояние между центрами:

$$R^c(W, S) = \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|} \right);$$

$$\alpha_U = \frac{|U|}{|W|}, \quad \alpha_V = \frac{|V|}{|W|},$$

$$\beta = -\alpha_U \alpha_V, \quad \gamma = 0.$$



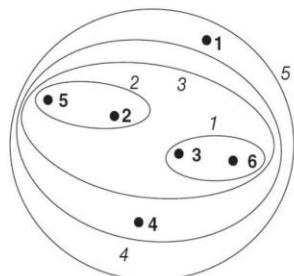
5. Расстояние Уорда:

$$R^y(W, S) = \frac{|S||W|}{|S|+|W|} \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|} \right);$$

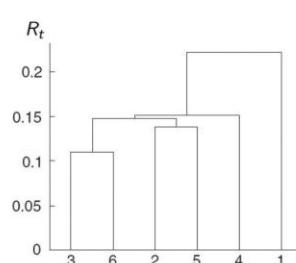
$$\alpha_U = \frac{|S|+|U|}{|S|+|W|}, \quad \alpha_V = \frac{|S|+|V|}{|S|+|W|}, \quad \beta = \frac{-|S|}{|S|+|W|}, \quad \gamma = 0.$$

1. Расстояние ближнего соседа:

Диаграмма вложения

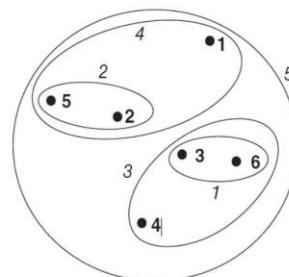


Дендрограмма

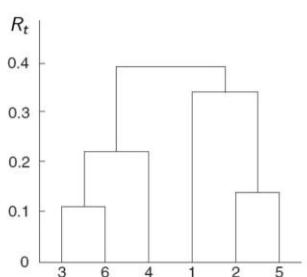


2. Расстояние дальнего соседа:

Диаграмма вложения

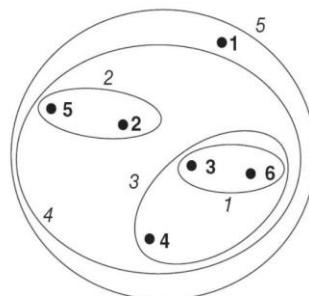


Дендрограмма

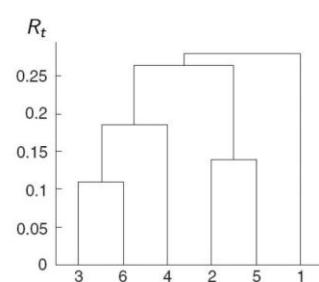


3. Групповое среднее расстояние:

Диаграмма вложения



Дендрограмма



Кластеризация

Иерархические алгоритмы

Свойство монотонности

Определение:

Кластеризация монотонна, если при каждом слиянии расстояние между объединяемыми кластерами только увеличивается: $R_1 < R_2 < R_3 \dots < R_l$

Теорема 7.1 (Миллиган, 1979). Если выполняются следующие три условия, то кластеризация является монотонной:

- 1) $\alpha_U \geq 0, \alpha_V \geq 0;$
- 2) $\alpha_U + \alpha_V + \beta \geq 1;$
- 3) $\min\{\alpha_U, \alpha_V\} + \gamma \geq 0.$

Из перечисленных выше расстояний только R^u не является монотонным. Расстояние Уорда отличается от него мультипликативной поправкой, которая и делает его монотонным.

Свойство монотонности

- Растворение — увеличивается расстояние до других кластеров (R_d и R_y) при росте t
- Сжатие — уменьшается расстояние до других кластеров при росте t (R_b)
- Сохраняющее метрику пространства — не изменяющие расстояние (R_c и $R_{\text{Ц}}$)

Проблема эффективности

- самая трудоёмкая операция в алгоритме Ланса-Уильямса — поиск ближайших кластеров — $O(\ell^2)$ операций:

$$\text{шаг 3: } (U, V) := \arg \min_{U \neq V} R(U, V).$$

- значит, построение всего дерева — $O(\ell^3)$ операций.

Идея повышения эффективности:

- перебирать лишь наиболее близкие пары:

$$\text{шаг 3: } (U, V) := \arg \min_{R(U, V) \leq \delta} R(U, V).$$

- периодически увеличивать параметр δ .

Кластеризация

Иерархические алгоритмы

Свойство редуктивности

Опр. (Брюнош, 1978). Расстояние R называется редуктивным, если для любого $\delta > 0$ и любых δ -близких кластеров U и V объединение δ -окрестностей U и V содержит в себе δ -окрестность кластера $W = U \cup V$:

$$\{S \mid R(U \cup V, S) < \delta, R(U, V) \leq \delta\} \subseteq \{S \mid R(S, U) < \delta \text{ или } R(S, V) < \delta\}.$$

Теорема (Диде и Моро, 1984). Если выполняются следующие три условия, то расстояние R является редуктивным:

- 1) $\alpha_U \geq 0, \alpha_V \geq 0$;
- 2) $\alpha_U + \alpha_V + \min\{\beta, 0\} \geq 1$;
- 3) $\min\{\alpha_U, \alpha_V\} + \gamma \geq 0$.

Резюме

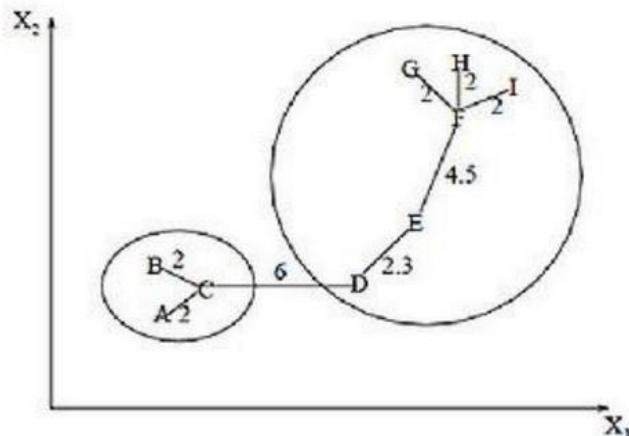
- Метод ближнего соседа — эффект «щупалец»
- Метод дальнего соседа — нелогичная кластеризация на ранних этапах
- Метод центра масс — не монотонен и нередуктивен
- Метод Уорда — показывает наилучшие экспериментальные результаты

Кластеризация

Послойная кластеризация

Алгоритмы, основанные на теории графов

- Выборка объектов представляется в виде графа $G=(V,E)$



Алгоритм послойной кластеризации

- Основан на выделении связных компонент графа на некотором уровне расстояний между объектами (вершинами)
- С-порог расстояния

$$0 \leq \rho(x, x') \leq 1 \quad 0 \leq c \leq 1$$

- Формирует последовательность подграфов графа G , которые отражают иерархические связи между кластерами

$$G^0 \subseteq G^1 \subseteq \dots \subseteq G^m$$

- $G^t = (V, E^t)$ — граф на уровне c^t

$$E^t = \{e_{ij} \in E : \rho_{ij} \leq c_t\}$$

- c^t — t -ый порог расстояния
- m — количество уровней иерархии
- $G^0 = (V, \emptyset)$, \emptyset — пустое множество ребер графа, получаемое при $c^0 = 0$
- $G^m = G$, то есть граф объектов без ограничений на расстояние (длину ребер графа), поскольку $c^m = 1$

Кластеризация

Послойная кластеризация

- Посредством изменения порогов расстояния $\{c^0, \dots, c^m\}$, где $0 = c^0 < c^1 < \dots < c^m = 1$, возможно контролировать глубину иерархии получаемых кластеров
- Вычислительная сложность: $O(\max(n, m))$, где $m < n(n-1)/2$
- Входные данные: последовательность порогов расстояния
- Результат: древовидная структура кластеров с разными уровнями иерархии

Практика

- Даны объекты:

$$\begin{pmatrix} 1 \\ 5 \\ 4 \\ 7 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \\ 5 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 9 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \\ 3 \\ 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 6 \\ 7 \\ 8 \\ 9 \\ 0 \end{pmatrix}$$

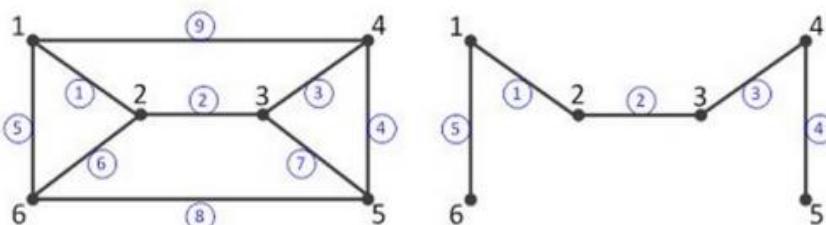
- В качестве расстояния использовать расстояние Чебышева: $l_\infty(\vec{x}, \vec{y}) = \max_{i=1, \dots, n} |x_i - y_i|$
- C1= 4, C2=5
- Представить решение в виде графа

Кластеризация

Алгоритмы поиска минимального покрывающего дерева

Определения

- Остовное дерево - ациклический связный подграф связного неориентированного графа, в который входят все его вершины.
- Минимальное остовное дерево - остовное дерево связного взвешенного неориентированного графа, имеющее минимальный вес.



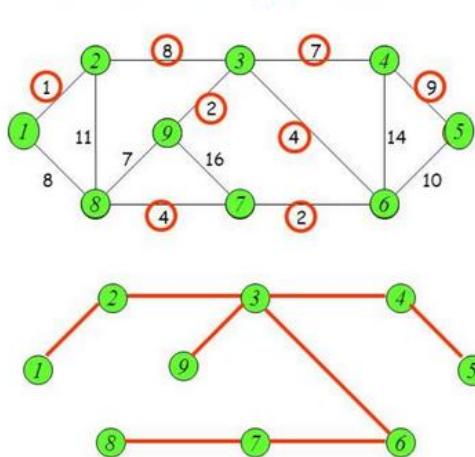
Алгоритм Прима

Пусть дан взвешенный неориентированный граф.

Для построения минимального остовного дерева необходимо:

1. Представить граф в виде матрицы смежности
2. Найти в матрице наименьший элемент, соответствующий ребру, соединяющему i -ю и j -ю вершины графа
3. Вычеркнуть элементы i -й и j -й строки матрицы
4. Пометить i -й и j -й столбцы матрицы
5. В помеченных столбцах i и j найти наименьший элемент, отличный от уже найденного
6. Повторять пункты 3-5 до тех пор, пока не будут задействованы все вершины графа

Prim's Algorithm



t [1:n-1, 1:2]	
1	2
2	3
3	9
.	.
6	7
7	8
3	4
4	5

Кластеризация

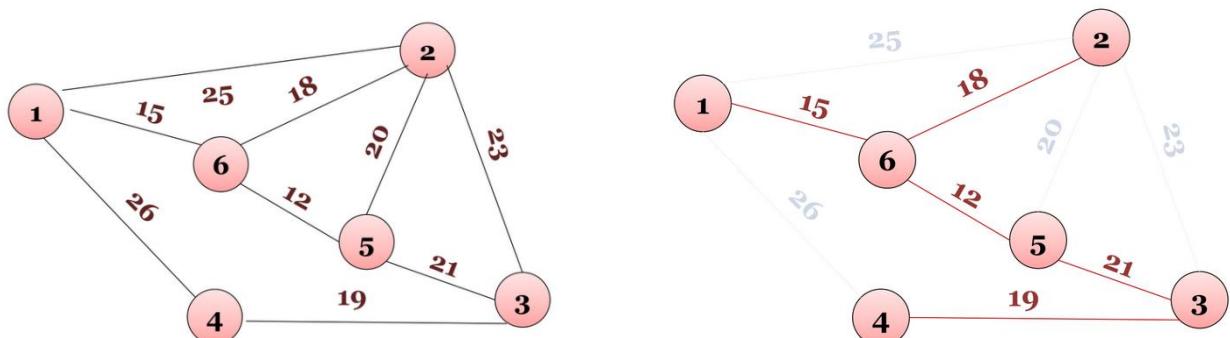
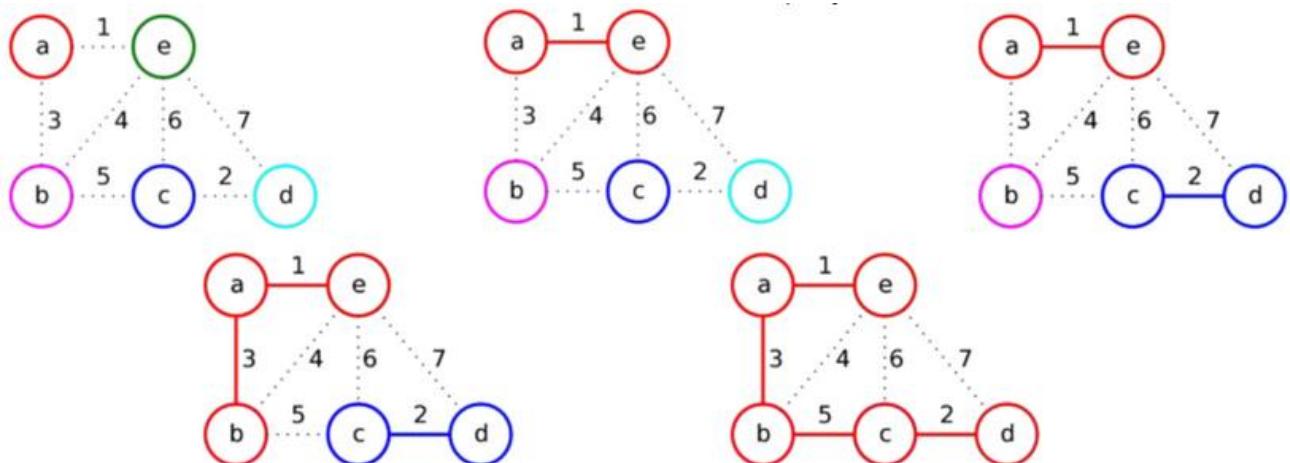
Алгоритмы поиска минимального покрывающего дерева

Алгоритм Краскала

Имеем два графа : исходный G и искомый K.
'K' получаем путем удаления из G всех ребер.

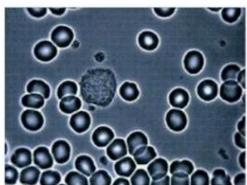
1. Сортируем ребра.
2. Один раз проходим по списку. Добавляем ребро из списка в множество ребер графа K, в случае, если в K после добавления не образуется циклов.

Ребра	ae	cd	ab	be	bc	ec	ed
Веса ребер	1	2	3	4	5	6	7



Выделение связных компонент

Примеры задач



Клетки крови



Ложки и сахар

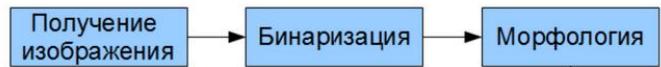


Монеты и купюры



Номера

Распознавание образов

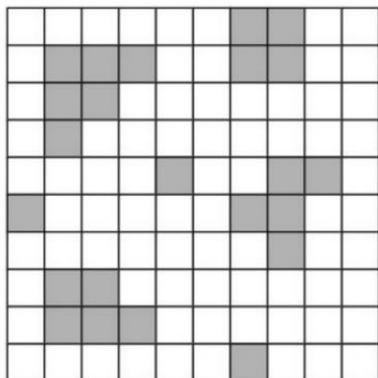


Получили бинарное изображение

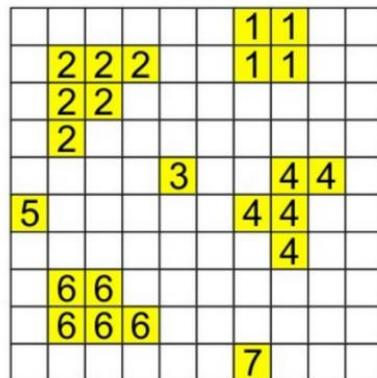


Нужна карта разметки

Разметка связных областей

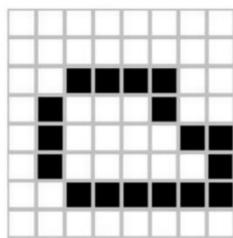
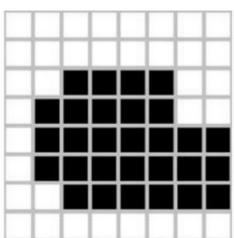


Бинарное изображение



Размеченное изображение

Связность



Виды связности

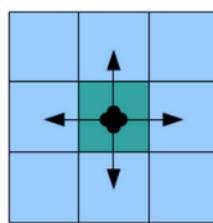


Рис. 1-а)

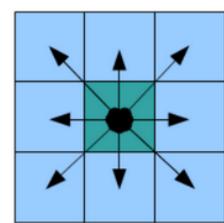


Рис. 1-б)

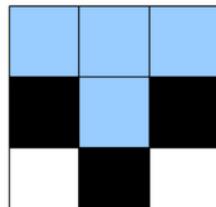


Рис. 2-а)

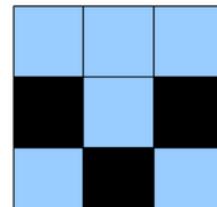


Рис. 2-б)

Выделение связных компонент

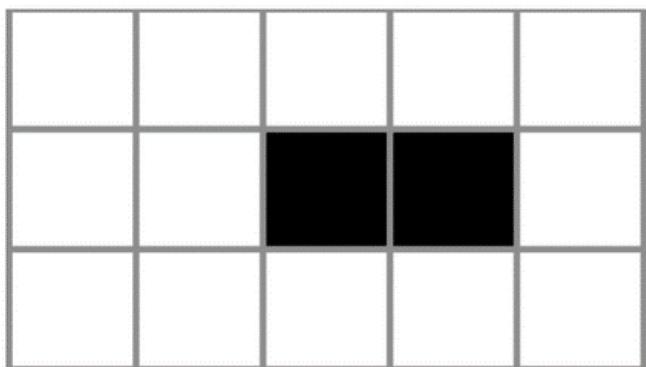
Рекурсивный алгоритм

- просмотр с точки (1,1);
- поочередно все точки строки: если все белые, переходим к следующей строке; иначе:
 - а) помечается точка;
 - б) определяется цвет четырех соседних точек;
 - в) если цвет всех точек белый или все помечены, закончить, иначе начальная позиция перемещается в первую найденную соседнюю черную точку;
 - г) повторяются пункты а, б и в;
- повторить до последней точки последней строки изображения.

```
void Labeling(BIT* img[], int* labels[])
{
    int L = 1;
    for(int y = 0; y < H; ++y)
        for(int x = 0; x < W; ++x)
            Fill(img, labels, x, y, L++);
}

void Fill(BIT* img[], int* labels[], int x, int y, int L)
{
    if( (labels[x][y] == 0) & (img[x][y] == 1) )
    {
        labels[x][y] = L;
        if( x > 0 )
            Fill(img, labels, x - 1, y, L);
        if( x < W - 1 )
            Fill(img, labels, x + 1, y, L);
        if( y > 0 )
            Fill(img, labels, x, y - 1, L);
        if( y < H - 1 )
            Fill(img, labels, x, y + 1, L);
    }
}
```

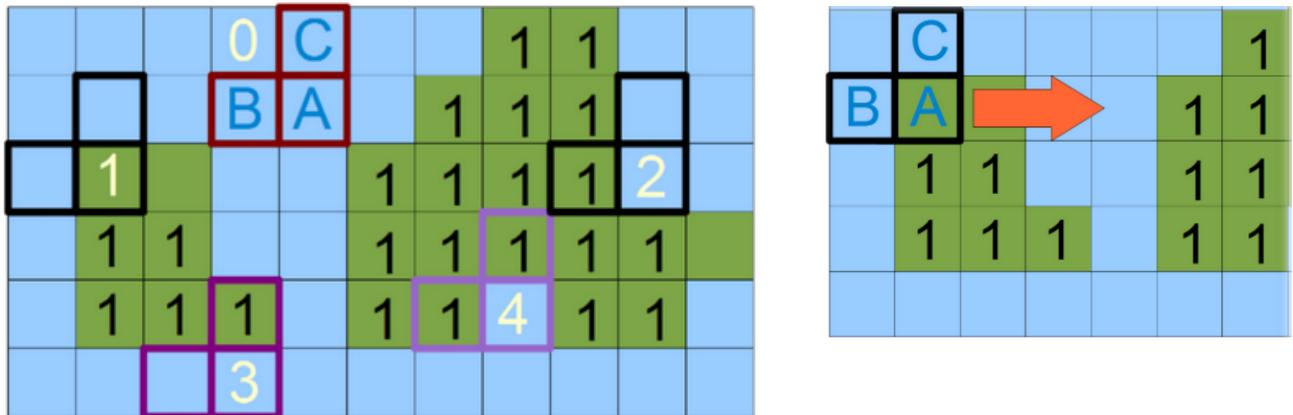
Пример



- (2;2) – белый;
- (2;3) – черный, не помечен:
 - Помечаем;
 - Рассмотрим соседей:
 - (1;3) – белый;
 - (2;2) – белый;
 - (2;4) – черный, не помечен:
 - Помечаем;
 - Рассмотрим соседей:
 - (1;4) – белый;
 - (2;3) – черный, помечен;
 - (2;5) – белый;
 - (3;4) – белый;
 - (3;3) – белый;
 - (2;4) – черный, помечен;
 - Конец.

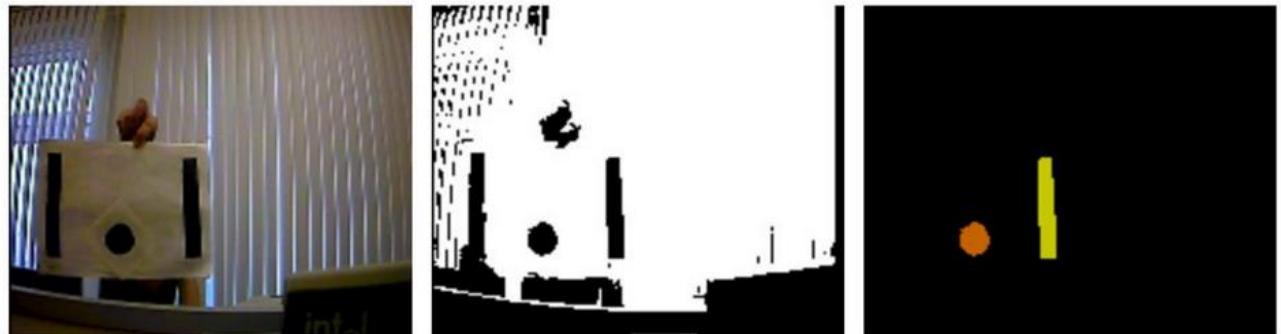
Выделение связных компонент

Итеративный алгоритм



- 0: пропускаем пиксель.
- 1: создание нового объекта — новый номер.
- 2: помечаем текущий пиксель А меткой, расположенной в В.
- 3: помечаем текущий пиксель А меткой, расположенной в С.
- 4: пиксель А может быть помечен либо как В либо как С.

Сведение к бинарному изображению



Выделение связных компонент

Алгоритмы, основанные на поиске границ областей



Общая схема работы данной группы методов

- Обнаружение границ регионов.
- Обнаружение регионов исходя из найденных границ.

Достоинства

- более гибок, чем простая бинаризация
- более стабилен к изменениям характеристик изображения

Недостатки

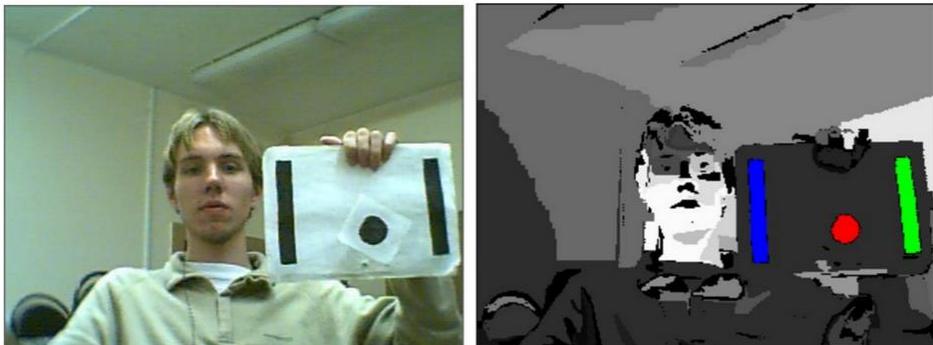
- Слабая устойчивость (достаточно небольшого разрыва границы для несрабатывания алгоритма)
- Выделение областей не является точным (часть региона "отьедается" границей)
- Скорость работы методов невысока

Алгоритмы, основанные на поиске регионов

- Объявляем левый верхний пиксель изображения новым классом.
- Для пикселей первой строки считаем отклонение от класса левого пикселя и, сравнивая с порогом, либо добавляем пиксель к классу соседа, либо заводим новый класс.
- Первый пиксель каждой последующей строки сравниваем с классом верхнего пикселя. Далее будем сравнивать пиксель с классами двух соседей: левого и верхнего.
 - Если отклонение от обоих классов больше порога, то заводим новый класс, если отклонение больше только для одного класса, то добавляем пиксель к тому классу отклонение от которого меньше порога.
 - Если отклонение допустимо для обоих классов, то возможно 2 варианта:
 - $L(g(C_1) - g(C_2)) < \delta$ - тогда объединяем эти 2 класса (если они не один и тот же класс) и добавляем к объединённому классу текущий пиксель.
 - $L(g(C_1) - g(C_2)) > \delta$ - в этом случае добавляем пиксель к тому классу, от которого отклонение минимально.

Выделение связных компонент

Алгоритмы, основанные на поиске регионов



Достоинства

- Широкий спектр применения
- Гибкость (можно эффективно менять чувствительность алгоритма)
- Скорость (быстрей алгоритма с поиском границы метода)
- Устойчивость (более устойчив к ошибкам)
- Точность (найденные регионы не «обкусаны»)

Точечные особенности

Определения

- **Точечная особенность изображения** m - это такая точка изображения, окрестность которой $\sigma(m)$ можно отличить от окрестности любой другой точки изображения $\sigma(n)$ в некоторой другой окрестности особой точки $\sigma_2(m)$.
- **Точечная особенность сцены** - такая точка сцены M , лежащая на плоском участке поверхности сцены $PlaneSegment$, изображение окрестности $I(PlaneSegment)$ которой можно отличить от изображений окрестностей всех других точек сцены N из некоторой другой окрестности этой точки $O(M)$.
- Точечной особенности сцены должна соответствовать точечная особенности изображений. Обратное неверно: существуют такие особые точки изображения, которым не соответствует никакие особые точки сцены. Такие точки называются ложными особенностями сцены (*false feature point*).

Задача слежения

Дана последовательность изображений $ImageSequence$ одной и той же сцены S , полученная с движущейся или неподвижной камеры, и набор точечных особенностей $\{N\}$, выделенных в первом кадре последовательности. Для каждой точечной особенности n из $\{N\}$ найти такие точки $n(t)$ на всех изображениях, что их окрестности будут максимально близки к окрестности $n(0)$, с учетом предполагаемой природы искажения ее окрестности и движения точки.

Основные допущения:

- “Малые” изменения от кадра к кадру
- Освещение от кадра к кадру - не меняется
- Особенность - плоская
- Изменения сцены описываются афинными преобразованиями

Точечные особенности

Задача слежения

Схема А

Этап 1 (детектирование)

- Определить в первом кадре особенности изображения

Этап 2 (слежение)

- Для каждого последующего кадра:
 - Для каждой особенности Feature(i):
 - Найти новое положение особенности в кадре t

Схема В

Этап 1 (детектирование и оценка)

- Найти набор особенностей {Features}
- Определить качество всех особенностей - Quality({Features})
- Оставить только особенности, чье качество выше некоторого заранее или динамически определенного порога, получив множество {GoodFeatures}

Этап 2 (слежение и оценка)

Для каждого последующего кадра:

- Найти в текущем кадре новое положение всех особенностей из {GoodFeature} - слежение (tracking)
- Определить текущее качество всех {GoodFeatures}
- Оставить только те особенности, чье качество удовлетворяет некоторому критерию
- Если число отслеживаемых точек падает ниже требуемого, применить детектор к текущему изображению и добавить в {GoodFeatures} новые точки.

Точечные особенности

Задача слежения

Нахождение набора особенностей

Рассмотрим матрицу:

$$\mathbf{M} = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

Если оба ее собственных значения велики, то даже небольшое смещение точки (x, y) в сторону вызывает значительные изменения в яркости. Что и соответствует особенности изображения.

Функция отклика угла записывается в следующем виде:

$$R = \det \mathbf{M} - k(\text{traceM})^2$$

Для снижения влияния шумов на найденные особенности используется сглаживание по Гауссу, но не в самом изображении, а в картах частных производных:

$$\left(\frac{\partial I}{\partial x}\right)^2, \left(\frac{\partial I}{\partial y}\right)^2, \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right)$$

Во многих случаях находится чересчур большое количество углов, из-за чего в дальнейшем их будет сложно отслеживать.

Поэтому вводится ограничение на минимальное расстояние между найденными особенностями, и все лишние отбрасываются.

Математическая формулировка задачи слежения

Пусть $I(x, t)$ - яркость изображения-кадра со временем t в точке x , где x - вектор. Движение изображения (image motion), вдали от границ видимости (occluding boundaries), описывается с помощью уравнения вида: $I(x, t) = I(\delta(x), t+t_1)$ (*), где $\delta(x)$ - движение точки x при переходе от кадра (t) к $(t+t_1)$.

Перемещение особенности от кадра к кадру описывается этим уравнением для всех точек x из окрестности особенности W .

Отметим, что в этом случае полагается, что освещение точки сцены, соответствующей особенности, остается постоянным.

При малых изменениях изображения от кадра к кадру можно считать, что окно особенности просто смещается, и движение $\delta(x)$ принимает вид $\delta(x) = x + d$. Однако при увеличении длительности слежения, изображение точки сцены искажается. Это искажение может быть приближенно описано аффинной трансформацией, поэтому движение точек описывается аффинным преобразованием $\delta(x) = Ax + d$, где A - матрица размерности 2×2 .

Задача трекера заключается в отыскании значения движения $\delta(x)$ для всех точек окна особенности W . Т.к. в реальных условиях (*) никогда строго не выполняется, то ищется такое движение, при котором минимизируется разница между окнами при текущем и будущем положении особенности, т.е. такое $\delta(x)$, при котором достигается минимум

$$e = |I(\delta(x), t+t_1) - I(x, t)|,$$

или

$$e = \min_W \sum (I(\delta(x), t+t_1) - I(x, t))^2$$

если норма разности изображений L_2

Точечные особенности

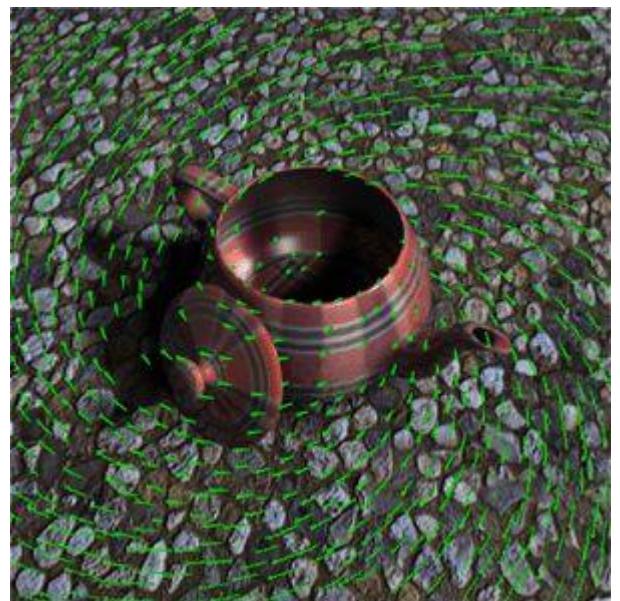
Задача слежения

Развитие алгоритмов слежения

Все современные алгоритмы слежения за особенностями опираются на работу 1981 году Лукаса и Канаде. В 1991 году математическая формулировка этого алгоритма была изменена, и стала основой для всех последующих обобщений с учетом аффинных искажений окрестности и освещенности. Путем замены соответствующих переменных на константы любой из них превращается в обычный алгоритм Lucas-Kanade.

- Lucas-Kanade - особенность считается только смещающейся, без искажений
- Tomasi-Kanade - переформулирование Lucas-Kanade. Движение считается смещением, и рассчитывается путем итеративного решения построенной системы линейных уравнений.
- Shi-Tomasi-Kanade - учитывает аффинные искажения особенности
- Jin-Favaro-Soatto - модификация Shi-Tomasi-Kanade с учетом аффинных изменений освещенности особенности

Пример трекинга особенностей



Классификация

Задача классификации

Классификация — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

В математической статистике задачи классификации называются также задачами дискриминантного анализа.

В машинном обучении задача классификации относится к разделу обучения с учителем. Существует также обучение без учителя, когда разделение объектов обучающей выборки на классы не задаётся, и требуется классифицировать объекты только на основе их сходства друг с другом. В этом случае принято говорить о задачах кластеризации или таксономии, и классы называть, соответственно, кластерами или таксонами.

Формальная постановка

Пусть X — множество описаний объектов, Y — конечное множество номеров (имён, меток) классов. Существует неизвестная целевая зависимость — отображение $y^*: X \rightarrow Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$. Требуется построить алгоритм $a: X \rightarrow Y$, способный классифицировать произвольный объект $x \in X$.

Классификация

Задача классификации

Вероятностная постановка задачи

Более общей считается вероятностная постановка задачи. Предполагается, что множество пар «объект, класс» $X \times Y$ является вероятностным пространством с неизвестной вероятностной мерой P . Имеется конечная обучающая выборка наблюдений $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$, сгенерированная согласно вероятностной мере P . Требуется построить алгоритм $a: X \rightarrow Y$, способный классифицировать произвольный объект $x \in X$.

Признаковое пространство

Признаком называется отображение $f: X \rightarrow D_f$, где D_f — множество допустимых значений признака. Если заданы признаки f_1, \dots, f_n , то вектор $x = (f_1(x), \dots, f_n(x))$ называется признаковым описанием объекта $x \in X$.

Признаковые описания допустимо отождествлять с самими объектами. При этом множество $X = D_{f_1} \times \dots \times D_{f_n}$ называют признаковым пространством.

В зависимости от множества D_f признаки делятся на следующие типы:

- бинарный признак: $D_f = \{0,1\}$;
- номинальный признак: D_f — конечное множество;
- порядковый признак: D_f — конечное упорядоченное множество;
- количественный признак: D_f — множество действительных чисел.

Часто встречаются прикладные задачи с разнотипными признаками, для их решения подходят далеко не все методы.

Классификация

Наивный байесовский классификатор

Теоретические основы

- Наивный байесовский алгоритм – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков.
- Теорема Байеса позволяет рассчитать апостериорную вероятность $P(c | x)$ на основе $P(c)$, $P(x)$ и $P(x | c)$.

Формулы

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

↑ ↑
Likelihood Class Prior Probability
↓ ↓
Posterior Probability Predictor Prior Probability

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

- **$P(c | x)$** – апостериорная вероятность данного класса с (т.е. данного значения целевой переменной) при данном значении признака x .
- **$P(c)$** – априорная вероятность данного класса.
- **$P(x | c)$** – правдоподобие, т.е. вероятность данного значения признака при данном классе.
- **$P(x)$** – априорная вероятность данного значения признака.

Пример

Рассмотрим обучающий набор данных, содержащий один признак «Погодные условия» (weather) и целевую переменную «Игра» (play), которая обозначает возможность проведения матча. На основе погодных условий мы должны определить, состоится ли матч. Чтобы сделать это, необходимо выполнить следующие шаги.

- **Шаг 1.** Преобразуем набор данных в частотную таблицу (frequency table).
- **Шаг 2.** Создадим таблицу правдоподобия (likelihood table), рассчитав соответствующие вероятности. Например, вероятность облачной погоды (overcast) составляет 0,29, а вероятность того, что матч состоится (yes) – 0,64.

Классификация

Наивный байесовский классификатор

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

- Шаг 3. С помощью теоремы Байеса рассчитаем апостериорную вероятность для каждого класса при данных погодных условиях. Класс с наибольшей апостериорной вероятностью будет результатом прогноза.

Задача.

Состоится ли матч при солнечной погоде (sunny)?

$$P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

$$P(\text{Sunny} \mid \text{Yes}) = 3 / 9 = 0,33$$

$$P(\text{Sunny}) = 5 / 14 = 0,36$$

$$P(\text{Yes}) = 9 / 14 = 0,64$$

$$P(\text{Yes} \mid \text{Sunny}) = 0,33 * 0,64 / 0,36 = 0,60$$

Значит, при солнечной погоде более вероятно, что матч состоится.

Положительные стороны

- Классификация, в том числе многоклассовая, выполняется легко и быстро.
- Когда допущение о независимости выполняется, НБА превосходит другие алгоритмы, такие как логистическая регрессия (logistic regression), и при этом требует меньший объем обучающих данных.
- НБА лучше работает с категорийными признаками, чем с непрерывными. Для непрерывных признаков предполагается нормальное распределение, что является достаточно сильным допущением.

Классификация

Наивный байесовский классификатор

Отрицательные стороны стороны

- Если в тестовом наборе данных присутствует некоторое значение категорийного признака, которое не встречалось в обучающем наборе данных, тогда модель присвоит нулевую вероятность этому значению и не сможет сделать прогноз. Это явление известно под названием «нулевая частота» (zero frequency). Данную проблему можно решить с помощью сглаживания. Одним из самых простых методов является сглаживание по Лапласу (Laplace smoothing).
- Хотя НБА является хорошим классификатором, значения спрогнозированных вероятностей не всегда являются достаточно точными. Поэтому не следует слишком полагаться на результаты, полученные методом НБА.
- Еще одним ограничением НБА является допущение о независимости признаков. В реальности наборы полностью независимых признаков встречаются крайне редко.

Приложения НБА

- **Классификация в режиме реального времени.** НБА очень быстро обучается, поэтому его можно использовать для обработки данных в режиме реального времени.
- **Многоклассовая классификация.** НБА обеспечивает возможность многоклассовой классификации. Это позволяет прогнозировать вероятности для множества значений целевой переменной.
- **Классификация текстов, фильтрация спама, анализ тональности текста.** При решении задач, связанных с классификацией текстов, НБА превосходит многие другие алгоритмы. Благодаря этому, данный алгоритм находит широкое применение в области фильтрации спама (идентификация спама в электронных письмах) и анализа тональности текста (анализ социальных медиа, идентификация позитивных и негативных мнений клиентов).
- **Рекомендательные системы.** Наивный байесовский классификатор в сочетании с колаборативной фильтрацией (collaborative filtering) позволяет реализовать рекомендательную систему. В рамках такой системы с помощью методов машинного обучения и интеллектуального анализа данных новая для пользователя информация отфильтровывается на основании спрогнозированного мнения этого пользователя о ней.

Классификация

Минимизация эмпирического риска

Основы

Метод минимизации эмпирического риска - это общий подход к решению широкого класса задач обучения по прецедентам

Задача обучения по прецедентам

- X - множество описаний объектов.
- Y - множество допустимых ответов.
- Существует неизвестная целевая зависимость $y^* : X \rightarrow Y$, значения которой известны только на объектах конечной обучающей выборки

$$X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}.$$

- Задача - построение алгоритма $a: X \rightarrow Y$, который бы приближал неизвестную целевую зависимость.

Функция потерь и эмпирический риск

Функция потерь - $\mathcal{L}(y, y')$, величина отклонения ответа $y = a(x)$ от правильного ответа $y' = y^*(x)$ на произвольном объекте $x \in X$.

Вводится модель алгоритмов $A = \{a: X \rightarrow Y\}$ в рамках которой будет вестись поиск отображения, приближающего неизвестную целевую зависимость.

Эмпирический риск — это функционал качества, характеризующий среднюю ошибку алгоритма a на выборке X^m :

$$Q(a, X^m) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(a(x_i), y^*(x_i)).$$

Минимизация эмпирического риска

Метод минимизации эмпирического риска заключается в том, чтобы в заданной модели алгоритмов A найти алгоритм, доставляющий минимальное значение функционалу эмпирического риска:

$$a = \arg \min_{a \in A} Q(a, X^m).$$

Разновидности функции потерь:

1. Классификация $\mathcal{L}(y, y') = [y' \neq y]$.
2. Регрессия $\mathcal{L}(y, y') = (y' - y)^2$.

Достоинства:

Конструктивный и универсальный подход, позволяющий сводить задачу обучения к задачам численной оптимизации.

Недостатки:

Переобучение, которое возникает практически всегда при использовании метода минимизации эмпирического риска.

Машинное обучение

- **Машинное обучение** — процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было явно заложено (запрограммировано).
- Говорят, что компьютерная программа обучается на основе опыта E по отношению к некоторому классу задач T и меры качества R , если качество решения задач из T , измеренное на основе R , улучшается с приобретением опыта
- На практике фаза обучения может предшествовать фазе работы алгоритма (например, детектирование лиц на снимке) — batch learning или обучение может проходить в процессе функционирования алгоритма (например, определение почтового спама) — online learning.

Классификация задач машинного обучения

- Дедуктивное обучение (экспертные системы)
- Индуктивное обучение (\approx статистическое обучение)
 - **Обучение с учителем:**
 - * классификация
 - * восстановление регрессии
 - * структурное обучение (structured learning)
 - * ...
 - **Обучение без учителя:**
 - * кластеризация
 - * визуализация данных
 - * понижение размерности
 - * ...
 - **Обучение с подкреплением (reinforcement learning)**
 - **Активное обучение**
 - ...

Аппарат

- Линейная алгебра
- Теория вероятностей и математическая статистика
- Методы оптимизации
- Численные методы
- Математический анализ
- Дискретная математика
- И др.

Машинное обучение

Сфера приложения

- Компьютерное зрение (computer vision)
- Распознавание речи (speech recognition)
- Компьютерная лингвистика и обработка естественных языков (natural language processing)
- Медицинская диагностика
- Биоинформатика
- Техническая диагностика
- Финансовые приложения
- Рубрикация, аннотирование и упрощение текстов
- Информационный поиск
- Интеллектуальные игры
- ...

Обучение по прецедентам

Множество \mathcal{X} – объекты, примеры (samples)

Множество \mathcal{Y} – ответы, отклики, «метки» (responses)

Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по $x \in \mathcal{X}$ предсказать (или оценить вероятность появления) $y \in \mathcal{Y}$.
(в частности, если зависимость детерминированная, то существует функция $f^* : \mathcal{X} \rightarrow \mathcal{Y}$)

Зависимость известна только на объектах из обучающей выборки:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$$

Пара $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$ – прецедент.

Задача обучения по прецедентам: восстановить зависимость, т. е. научиться по новым объектам $x \in \mathcal{X}$ предсказывать ответы $y \in \mathcal{Y}$.

Признаковые описания

$$x \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

где $Q_j = \mathbf{R}$ или Q_j – конечно

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X}$$

x_j – j -й признак (свойство, атрибут) объекта x .

Машинное обучение

Признаковые описания

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или фактор).
Можно считать, например, что, $Q_j = \{1, 2, \dots, s_j\}$.
Если $|Q_j| = 2$, то признак *бинарный* и можно считать, например, $Q_j = \{0, 1\}$ или $Q_j = \{-1, 1\}$.
- Если Q_j конечно и упорядочено, то признак *порядковый*.
Например, $Q = \{\text{Beginner}, \text{Elementary}, \text{Intermediate}, \text{Advanced}, \text{Proficiency}\}$ (уровень владения английским языком)
- Если $Q_j = \mathbb{R}$, то признак *количественный*.

Аналогично для выходов:

$y \in \mathcal{Y}$, где $\mathcal{Y} = \mathbb{R}$ или \mathcal{Y} — конечно.

x называется *входом*,

y — *выходом*, или *откликом*

Компоненты x_j вектора x так же называют *входами* или *предикатными (объясняющими) переменными*.

В мат. статистике x_j называют «независимыми» переменными, а y — «зависимой».

Входные переменные и соответствующие им выходы известны для объектов обучающей выборки.

Значения признаков объектов из обучающей выборке и соответствующие ответы обычно записывают в матрицы:

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \dots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}$$

$$(\mathbf{X} \mid \mathbf{y}) = \left(\begin{array}{cccc|c} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} & y^{(2)} \\ \dots & & & & \dots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} & y^{(N)} \end{array} \right)$$

Машинное обучение

Классификация задач обучения с учителем

В зависимости от множества \mathcal{Y} выделяют разные типы задачи обучения.

- \mathcal{Y} конечно, например, $\mathcal{Y} = \{1, 2, \dots, K\}$, — *задача классификации* (или *задача распознавания образов*):

\mathcal{X} разбивается на K классов

$$\mathcal{X}_k = \{x \in \mathcal{X} : f(x) = k\} \quad (k = 1, 2, \dots, K).$$

По x требуется предсказать, какому классу он принадлежит.

- $\mathcal{Y} = \mathbb{R}$ — *задача восстановления регрессии*.

Требуется найти функцию f из определенного класса, которая аппроксимирует неизвестную зависимость.

Ситуация, когда y — вектор, сводится к несколькими задачам со скалярным (атомарным) выходом.

y может быть чем-то более хитрым, например, графом, деревом, цепочкой символов (нефиксированной длины) — *структурное машинное обучение* (structured learning)

Обучение без учителя

Обучение по прецедентам — это *обучение с учителем*

Такое обучение можно рассматривать как игру двух лиц: ученика, который должен восстановить зависимость, и учителя, который для объектов из обучающей выборки указывает ученику соответствующий им выход.

Иногда можно считать, что объекты из обучающей выборки предъявляются средой, а иногда — их выбирает сам учитель, в некоторых случаях их выбирает ученик (*активное обучение*).

Рассматривается также *обучение без учителя*.

В этом случае нет учителя и «обучающая выборка» состоит только из объектов.

Ученик, имея только список объектов $x^{(1)}, x^{(2)}, \dots, x^{(N)}$, должен определить, как объекты связаны друг с другом.

Например, разбить объекты на группы (*кластеры*), так, чтобы в одном кластере оказались близкие друг к другу объекты, а в разных кластерах объекты были существенно различные.

Машинное обучение

Частичное обучение

- Каждый прецедент представляет собой пару «объект, ответ», но ответы известны только на части прецедентов.

Трансдуктивное обучение

- Дана конечная обучающая выборка прецедентов. Требуется по этим частным данным сделать предсказания относительно других частных данных — тестовой выборки. В отличие от стандартной постановки, здесь не требуется выявлять общую закономерность, поскольку известно, что новых тестовых прецедентов не будет. С другой стороны, появляется возможность улучшить качество предсказаний за счёт анализа всей тестовой выборки целиком, например, путём её кластеризации. Во многих приложениях трансдуктивное обучение практически не отличается от частичного обучения.

Обучение с подкреплением

- Роль объектов играют пары «ситуация, принятное решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды).

Обозначения

d число входных признаков

N длина обучающей выборки

\mathcal{X} множество объектов

\mathcal{Y} множество ответов (выходов)

$x^{(1)}, x^{(2)}, \dots, x^{(N)}$ объекты обучающей выборки, $x^{(i)} \in \mathcal{X}$ ($i = 1, 2, \dots, N$)

$y^{(1)}, y^{(2)}, \dots, y^{(N)}$ выходы для объектов из обучающей выборки, $y^{(i)} \in \mathcal{Y}$

K количество классов (в задачах классификации)

$\Pr A$ вероятность события A

$\Pr(A|B)$ вероятность события A при условии, что наступило событие B

$P_X(x)$ интегральная функция распределения: $P_X(x) = \Pr\{X \leq x\}$

$p_X(x)$ плотность вероятности непрерывной случайной величины X

$P(y|x)$ условная интегральная функция распределения

$p(y|x)$ условная плотность вероятности

$E X$ математическое ожидание случайной величины X

$D X$ или $\text{Var } X$ дисперсия случайной величины X

σX среднее квадратическое отклонение: $\sigma X = \sqrt{D X}$

Машинное обучение

Вероятностная постановка задачи

$\mathcal{X} = \mathbf{R}^d$ — множество объектов (входов) (точнее: множество их описаний)

$\mathcal{Y} = \mathbf{R}$ — множество ответов (выходов)

Будем рассматривать пары (x, y) как реализации $(d + 1)$ -мерной случайной величины (X, Y) , заданной на вероятностном пространстве

$$(\mathcal{X} \times \mathcal{Y}, \mathbf{A}, \Pr), \quad X \in \mathbf{R}^d, Y \in \mathbf{R}.$$

j -й признак — бинарный, номинальный или порядковый $\Leftrightarrow X_j$ — дискретная с. в.

j -й признак — количественный $\Leftrightarrow X_j$ — непрерывная с. в.

Интегральный закон распределения $P_{X,Y}(x, y)$ не известен, однако известна *обучающая выборка*

$$\left\{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)}) \right\},$$

где $(x^{(i)}, y^{(i)})$ являются независимыми реализациями случайной величины (X, Y) .

Требуется найти функцию $f : \mathcal{X} \rightarrow \mathcal{Y}$, которая по x предсказывает y , $f \in \mathcal{F}$

f называется *решающей функцией* или *решающим правилом*, а также *классификатором* (в случае задачи классификации).

Построение f называют *обучением, настройкой модели* и т. п.

Метод ближайших соседей

Будем, как и в задаче восстановления регрессии, для аппроксимации $\Pr(y|x)$ использовать k ближайших (по некоторому, например, евклидову расстоянию) объектов из обучающей выборки. Получаем метод k ближайших соседей для задачи классификации.

Пусть $N_k(x)$ — множество из k ближайших к x (по евклидову расстоянию) точек из обучающей выборки,

$I_k(x, y)$ — множество тех точек $x^{(i)}$ из $N_k(x)$, для которых $y^{(i)} = y$.

Согласно *методу k ближайших соседей* (k NN — k nearest neighbours) в качестве $f(x)$ берем результат голосования по всем точкам из $I_k(x, y)$:

$$f(x) = \operatorname{argmax}_y |I_k(x, y)|,$$

Частным случаем является *метод (одного) ближайшего соседа*, в котором $f(x) = y^{(i)}$, где $x^{(i)}$ — ближайший к x объект из обучающей выборки.

В этом случае D_y представляют собой *области Вороного*

Деревья принятия решений

Дерево

Дерево — одна из наиболее широко распространённых структур данных в информатике, эмулирующая древовидную структуру в виде набора связанных узлов. Является связанным графом, не содержащим циклы.

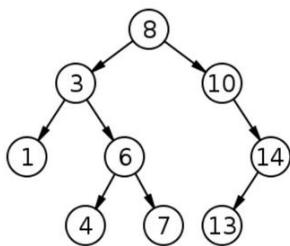


Рис.1. Пример дерева

- *Корневой узел* — самый верхний узел дерева (узел 8);
- *Лист* — узел, не имеющий дочерних элементов (узлы 1, 4, 7, 13);
- *Внутренний узел* — любой узел дерева, имеющий потомков(3, 6, 10, 14).

Дерево принятия решений

- Дерево принятия решений — это дерево, в листьях которого стоят значения целевой функции, а в остальных узлах — условия перехода, определяющие по какому из ребер идти.
- Если для данного наблюдения условие истинно(true), то осуществляется переход по левому ребру, если же ложно(false) — по правому.

Понятие энтропии

Энтропия означает меру неупорядоченности системы; чем меньше элементы системы подчинены какому-либо порядку, тем выше энтропия.

Энтропия Шеннона:

$$S = - \sum p_i * \ln(p_i)$$

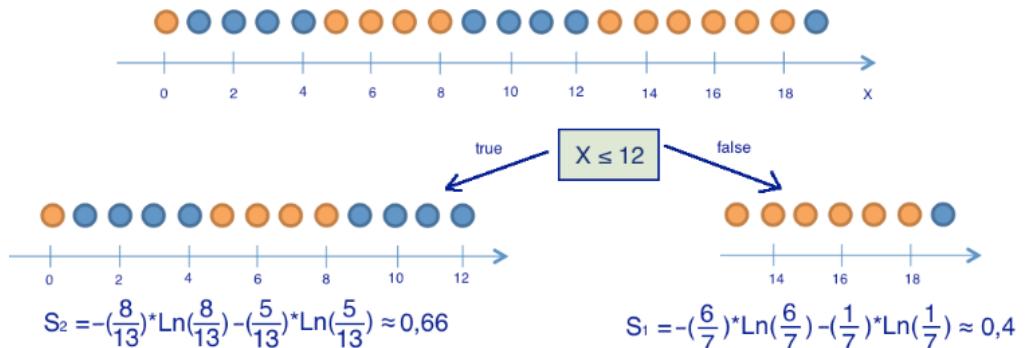
$$p_i = \frac{N_i}{N}$$

Деревья принятия решений

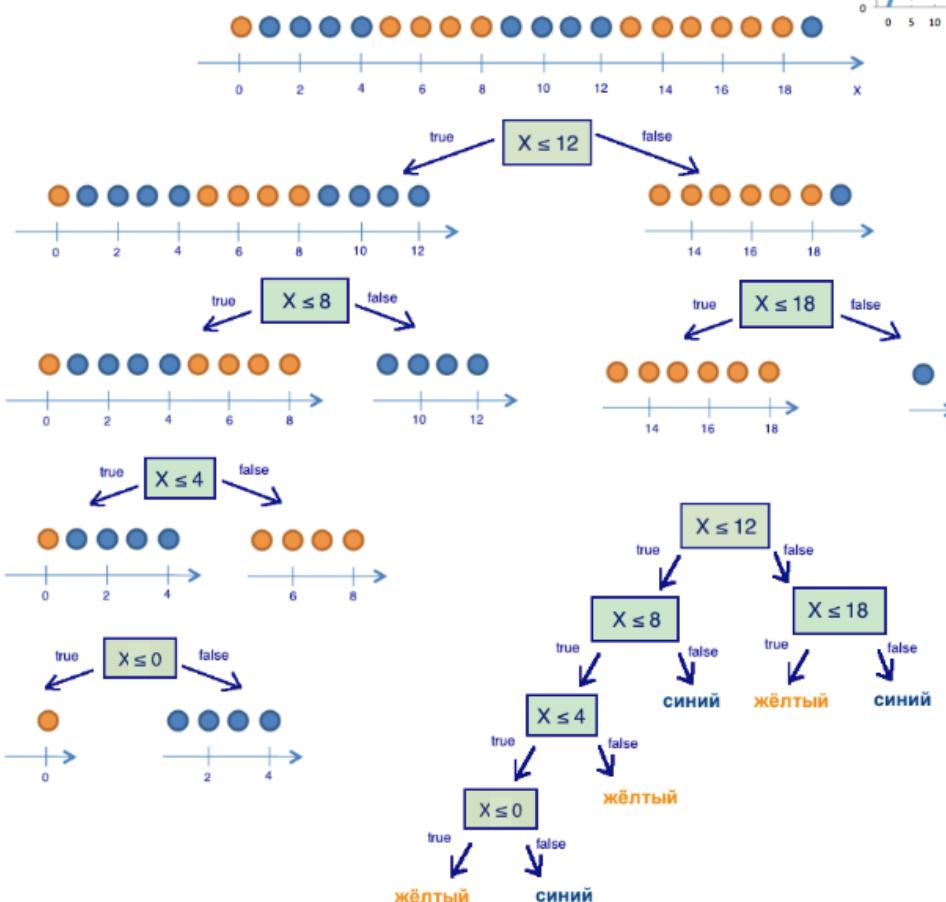
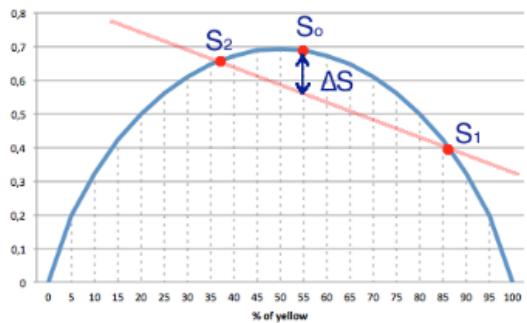
Пример

Рассмотрим множество двухцветных шариков, в котором цвет шарика зависит только от координаты x (при расчетах используем энтропию Шеннона):

$$S_0 = -\left(\frac{9}{20}\right) \cdot \ln\left(\frac{9}{20}\right) - \left(\frac{11}{20}\right) \cdot \ln\left(\frac{11}{20}\right) \approx 0,69$$



Вывод: если таким образом разделить множество на две части, то средняя энтропия будет меньше исходной на ΔS . Это значит, что данный предикат обобщает некоторую информацию о данных.



Деревья принятия решений

Алгоритм построения дерева принятия решений

s_0 = вычисляем энтропию исходного множества

Если $s_0 == 0$ значит:

Все объекты исходного набора, принадлежат к одному классу

Сохраняем этот класс в качестве листа дерева

Если $s_0 != 0$ значит:

Перебираем все элементы исходного множества:

Для каждого элемента перебираем все его атрибуты:

На основе каждого атрибута генерируем предикат, который разбивает исходное множество на два подмножества

Рассчитываем среднее значение энтропии

Вычисляем ΔS

Нас интересует предикат, с наибольшим значением ΔS

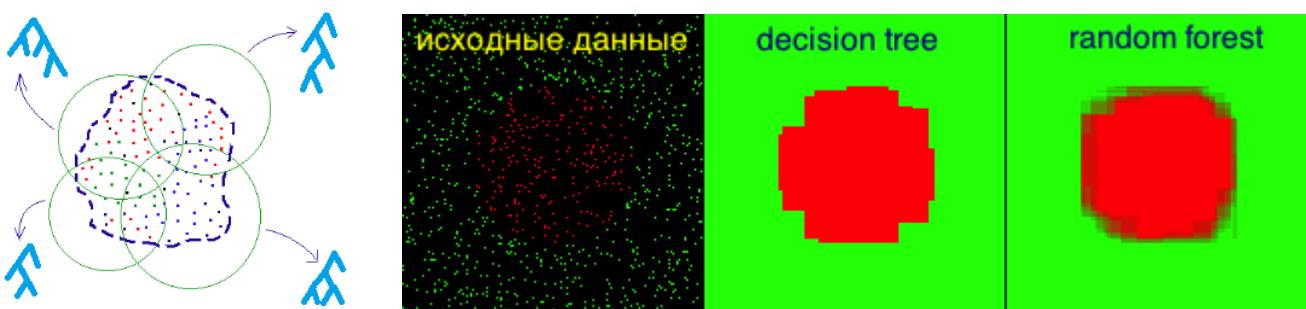
Найденный предикат является частью дерева принятия решений, сохраняем его

Разбиваем исходное множество на подмножества, согласно предикату

Повторяем данную процедуру рекурсивно для каждого подмножества

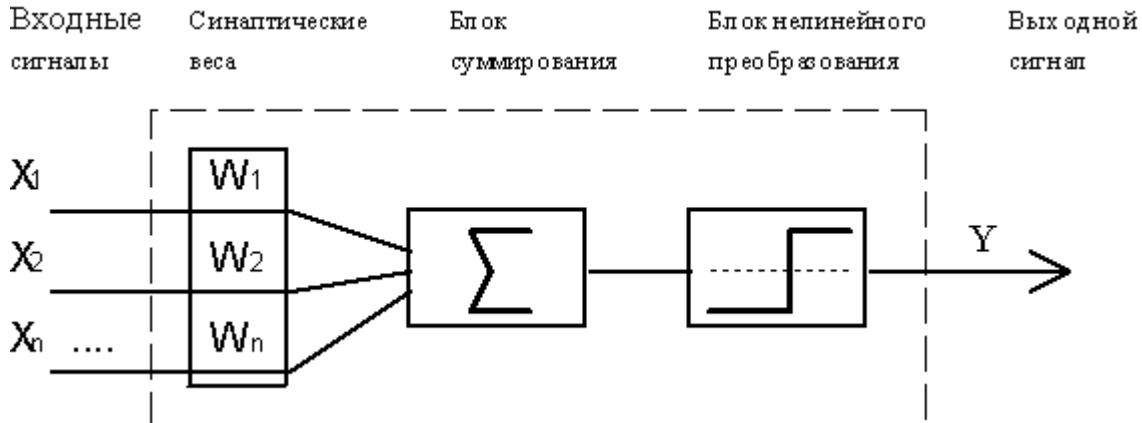
Random forest

- Выбираем случайные подмножества из обучающей выборки,
- Для каждого подмножества строим своё дерево принятия решений.
- Каждое дерево «голосует» за принадлежность объекта к определённому классу.
- Определяем с какой вероятностью объект принадлежит к какому-либо классу



Нейронные сети

Формальный нейрон



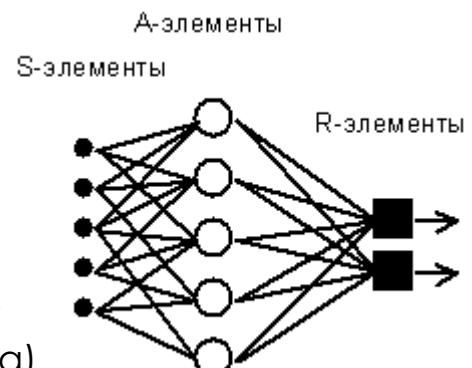
- Вектор локальной памяти содержит информацию о весовых множителях, с которыми входные сигналы будут интерпретироваться нейроном
- В блоке суммирования происходит накопление общего входного сигнала равного взвешенной сумме входов

$$net = \sum_{i=1}^n W_i x_i$$

- Отклик нейрон далее описывается по принципу "все или ничего", т. е. переменная подвергается нелинейному пороговому преобразованию, при котором выход (состояние активации нейрона) Y устанавливается равным единице, если $net > Q$, и $Y=0$ в обратном случае. Значение порога Q (часто полагаемое =нулю) также хранится в локальной памяти.

Персепtron Розенблатта

- S-элементы формируют сетчатку сенсорных клеток, принимающих двоичные сигналы от внешнего мира.
- Далее сигналы поступают в слой ассоциативных или А-элементов (для упрощения изображения часть связей от входных S-клеток к А-клеткам не показана).



Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей.

R-элементы с фиксированными весами формируют сигнал реакции персептрана на входной стимул.

Нейронные сети

Теорема об обучении персептрана

- Обучение сети состоит в подстройке весовых коэффициентов каждого нейрона
- Пусть имеется набор пар векторов (x^a, y^a) , $a = 1..p$, называемый обучающей выборкой
- Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора x^a на выходах всякий раз получается соответствующий вектор y^a

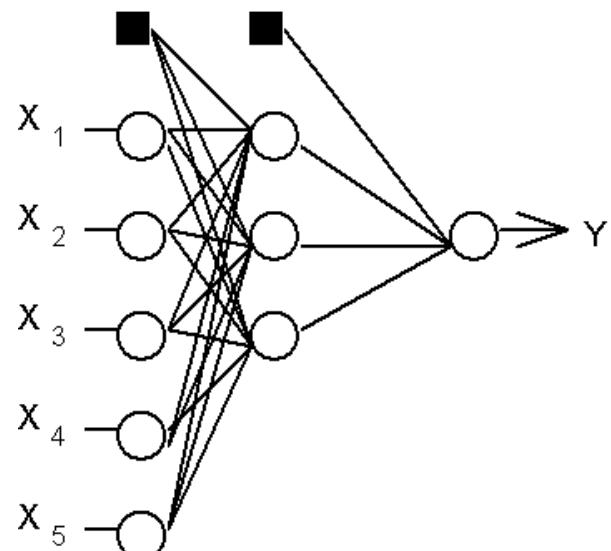
Алгоритм коррекции по ошибке

- Шаг 0: Начальные значения весов всех нейронов $W(t = 0)$ полагаются случайными
- Шаг 1: Сети предъявляется входной образ x^a , в результате формируется выходной образ $y^a \neq \hat{y}^a$
- Шаг 2: Вычисляется вектор ошибки $\delta^a = (y^a - \hat{y}^a)$, делаемой сетью на выходе. Дальнейшая идея состоит в том, что изменение вектора весовых коэффициентов в области малых ошибок должно быть пропорционально ошибке на выходе, и равно нулю если ошибка равна нулю
- Шаг 3: Вектор весов модифицируется по следующей формуле:
$$W(t + \Delta t) = W(t) + \eta x^a \cdot (\delta^a)^T$$
 Здесь $0 < \eta < 1$ - темп обучения
- Шаг 4: Шаги 1-3 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется эпохой. Обучение завершается по истечении нескольких эпох, а) когда итерации сойдутся, т.е. вектор весов перестает изменяться, или б) когда полная просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

Нейронные сети

Многослойный персепtron

- Межнейронные синаптические связи сети устроены таким образом, что каждый нейрон на данном уровне иерархии принимает и обрабатывает сигналы от каждого нейрона более низкого уровня. Таким образом, в данной сети имеется выделенное направление распространения нейроимпульсов – от входного слоя через один (или несколько) скрытых слоев к выходному слою нейронов
- Персептрон представляет собой сеть, состоящую из нескольких последовательно соединенных слоев формальных нейронов МакКаллока и Питтса
- На низшем уровне иерархии находится входной слой, состоящий из сенсорных элементов, задачей которого является только прием и распространение по сети входной информации
- Далее имеются один или, реже, несколько скрытых слоев. Каждый нейрон на скрытом слое имеет несколько входов, соединенных с выходами нейронов предыдущего слоя или непосредственно со входными сенсорами $X_1..X_n$, и один выход
- Нейрон характеризуется уникальным вектором весовых коэффициентов w . Веса всех нейронов слоя формируют матрицу, которую мы будем обозначать V или W
- Функция нейрона состоит в вычислении взвешенной суммы его входов с дальнейшим нелинейным преобразованием ее в выходной сигнал:



$$y = \frac{1}{1 + \exp(-[\sum_i W_i x_i - \Theta])}$$

Нейронные сети

Метод обратного распространения ошибок

- Для упрощения обозначений ограничимся ситуацией, когда сеть имеет только один скрытый слой
- Матрицу весовых коэффициентов от входов к скрытому слою обозначим W , а матрицу весов, соединяющих скрытый и выходной слой - как V
- Для индексов примем следующие обозначения: входы будем нумеровать только индексом i , элементы скрытого слоя – индексом j , а выходы, соответственно, индексом k
- Пусть сеть обучается на выборке (X_a, Y_a) , $a = 1..p$. Активности нейронов будем обозначать малыми буквами y с соответствующим индексом, а суммарные взвешенные входы нейронов - малыми буквами x
- Шаг 0:** Начальные значения весов всех нейронов всех слоев $V(t=0)$ и $W(t=0)$ полагаются случайными числами.
- Шаг 1:** Сети предъявляется входной образ X_a , в результате формируется выходной образ $y \neq Y_a$. При этом нейроны последовательно от слоя к слою функционируют по следующим формулам:

$$\text{скрытый слой } x_j = \sum_i W_{ij} X_i^\alpha; y_j = f(x_j)$$

$$\text{выходной слой } x_k = \sum_j V_{jk} y_j; y_k = f(x_k)$$

Здесь $f(x)$ - сигмоидальная функция

- Шаг 2:** Функционал квадратичной ошибки сети для данного входного образа имеет вид:
$$E = 1 / 2 \sum_k (y_k - Y_k^\alpha)^2$$
- Данный функционал подлежит минимизации. Классический градиентный метод оптимизации состоит в итерационном уточнении аргумента согласно формуле:

$$V_{jk}(t+1) = V_{jk}(t) - h \cdot \frac{\partial E}{\partial V_{jk}}$$

- Функция ошибки в явном виде не содержит зависимости от веса V_{jk} , поэтому воспользуемся формулами неявного дифференцирования сложной функции:

Нейронные сети

Метод обратного распространения ошибок

$$\frac{\partial E}{\partial y_k} = \delta_k = (y_k - Y_k^\alpha);$$

$$\frac{\partial E}{\partial x_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} = \delta_k \cdot y_k(1 - y_k)$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} \cdot \frac{\partial x_k}{\partial w_{jk}} =$$

$$= \delta_k \cdot y_k(1 - y_k) \cdot y_j$$

Здесь учтено полезное свойство сигмоидальной функции $f(x)$: ее производная выражается только через само значение функции, $f'(x)=f(1-f)$

- **Шаг 3:** На этом шаге выполняется подстройка весов скрытого слоя. Градиентный метод по-прежнему дает:

$$W_{ij}(t+1) = W_{ij}(t) - h \cdot \frac{\partial E}{\partial w_{ij}}$$

- Вычисления производных выполняются по тем же формулам, за исключением некоторого усложнения формулы для ошибки δ_j .

$$\frac{\partial E}{\partial x_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} = \delta_k \cdot y_k(1 - y_k)$$

$$\begin{aligned}\frac{\partial E}{\partial y_j} &= \delta_j = \sum_k \frac{\partial E}{\partial x_k} \cdot \frac{\partial x_k}{\partial y_j} = \\ &= \sum_k \delta_k \cdot y_k(1 - y_k) \cdot V_{jk};\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = \\ &= \delta_j \cdot y_j(1 - y_j) \cdot X_i^\alpha = \\ &= [\sum_k \delta_k \cdot y_k(1 - y_k) \cdot V_{jk}] \cdot [y_j(1 - y_j) \cdot X_i^\alpha]\end{aligned}$$

При вычислении δ_j здесь и был применен принцип обратного распространения ошибки: частные производные берутся только по переменным последующего слоя. По полученным формулам модифицируются веса нейронов скрытого слоя.

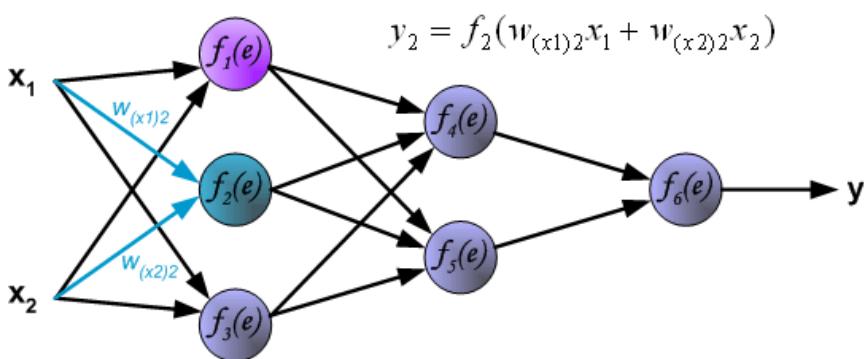
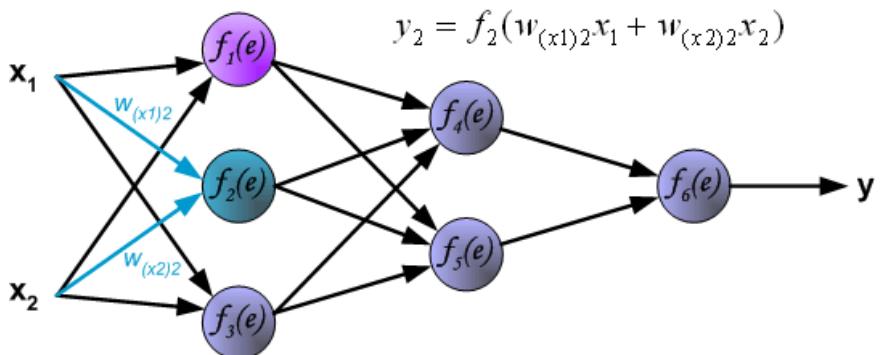
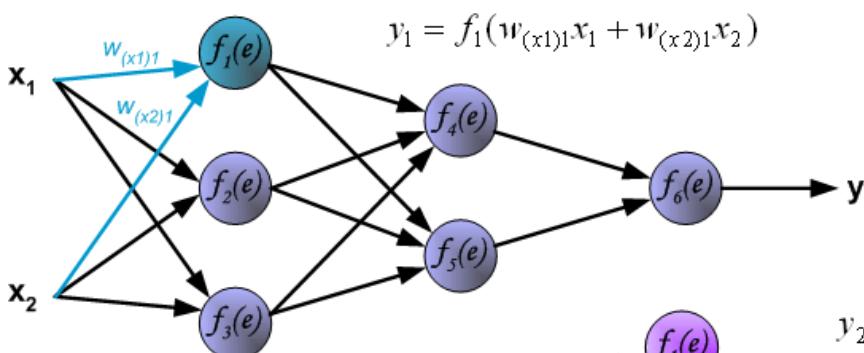
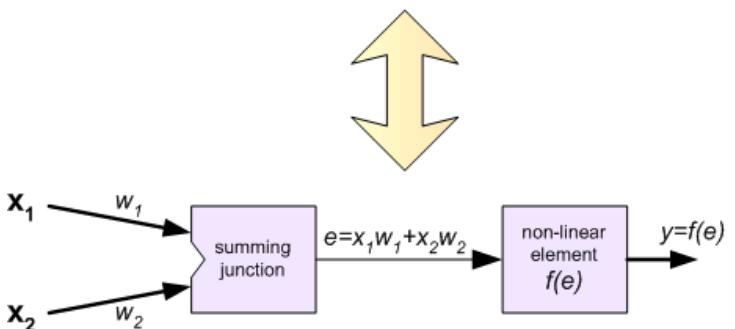
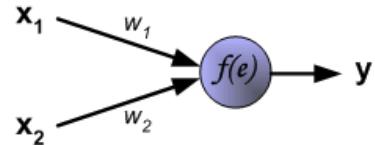
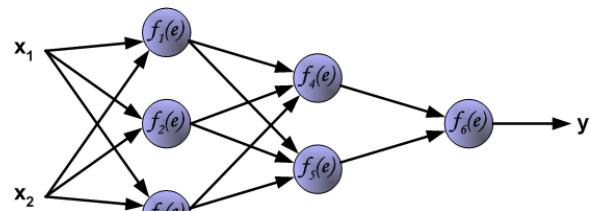
Если в нейронной сети имеется несколько скрытых слоев, процедура обратного распространения применяется последовательно для каждого из них, начиная со слоя, предшествующего выходному, и далее до слоя, следующего за входным. При этом формулы сохраняют свой вид с заменой элементов выходного слоя на элементы соответствующего скрытого слоя.

Нейронные сети

Метод обратного распространения ошибок

- Шаги 1-3 повторяются для всех обучающих векторов. Обучение завершается по достижении малой полной ошибки или максимально допустимого числа итераций, как и в методе обучения Розенблatta

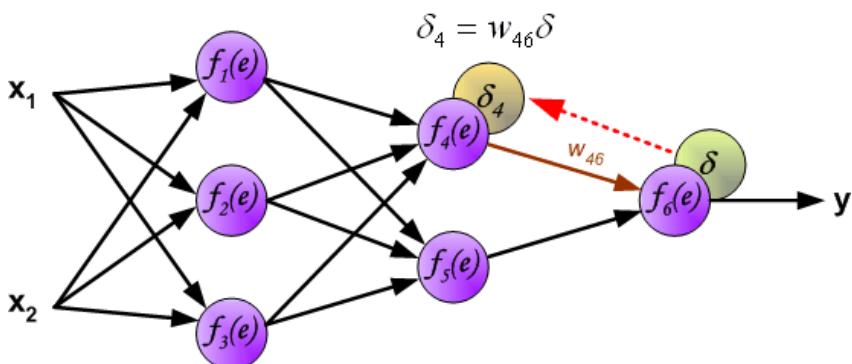
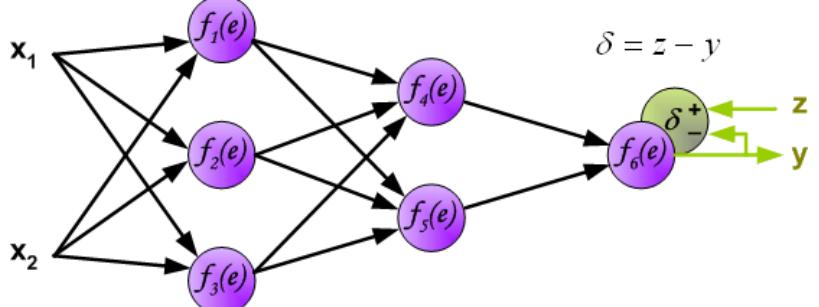
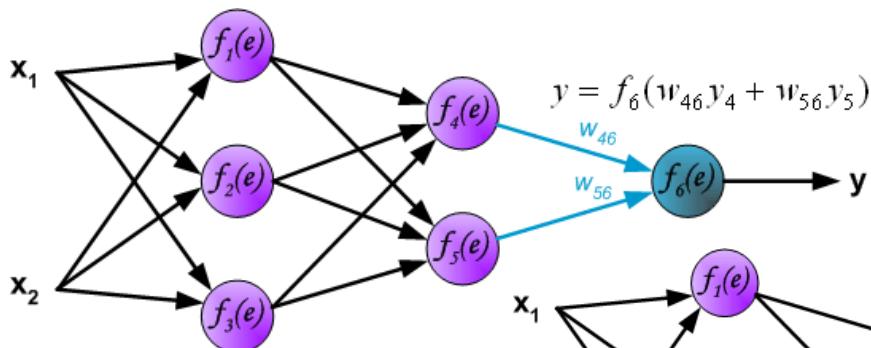
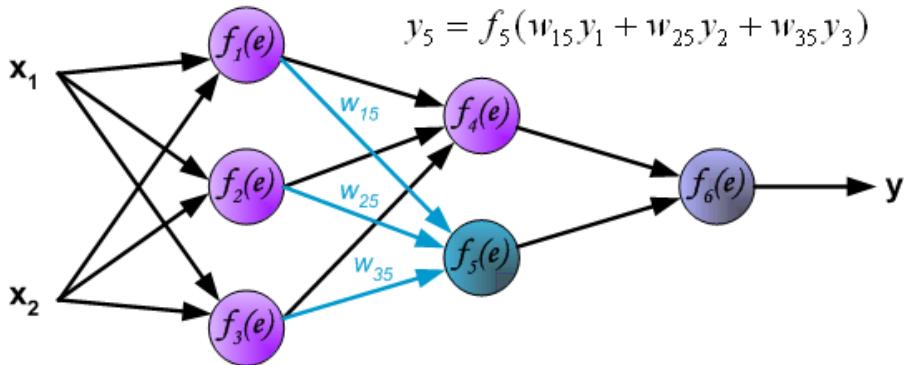
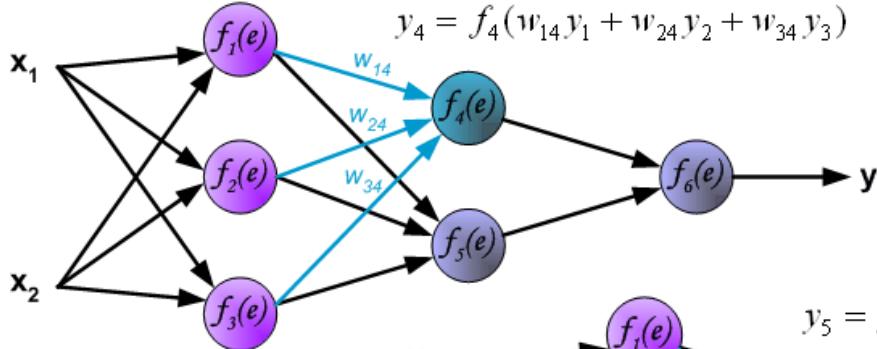
Пример



Нейронные сети

Метод обратного распространения ошибок

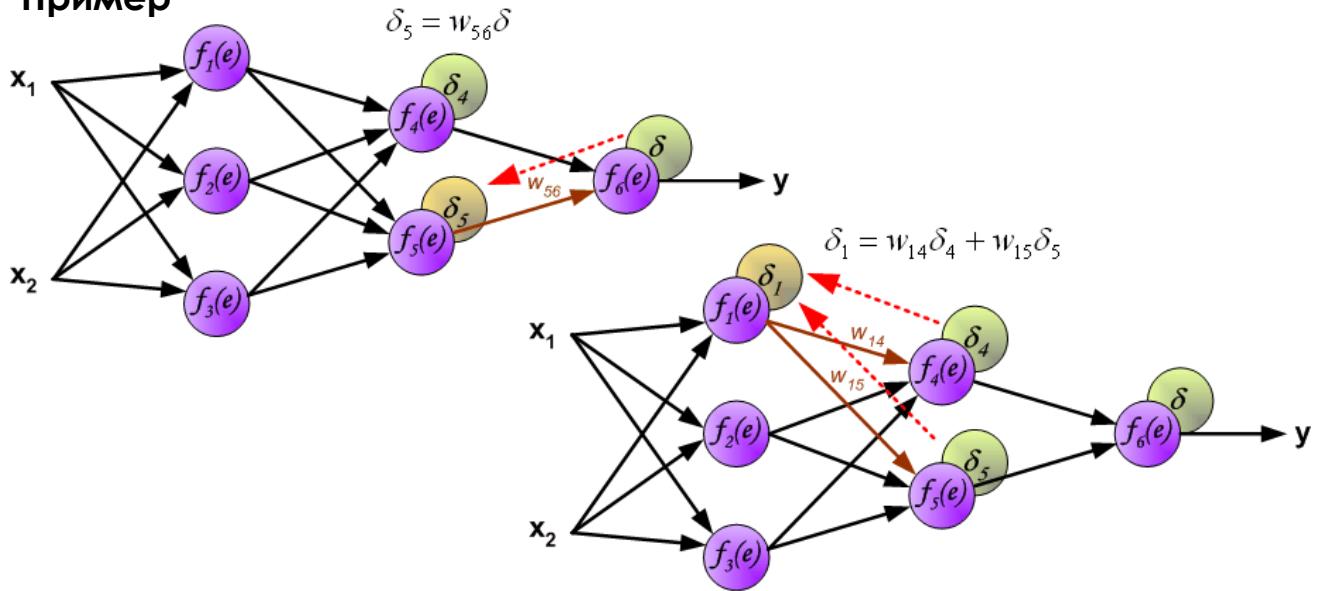
Пример



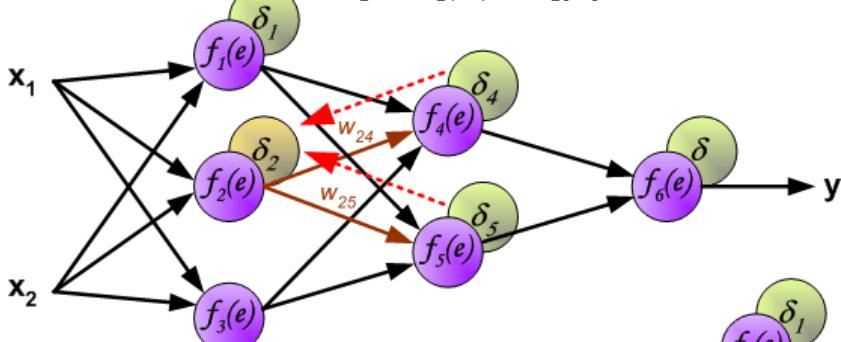
Нейронные сети

Метод обратного распространения ошибок

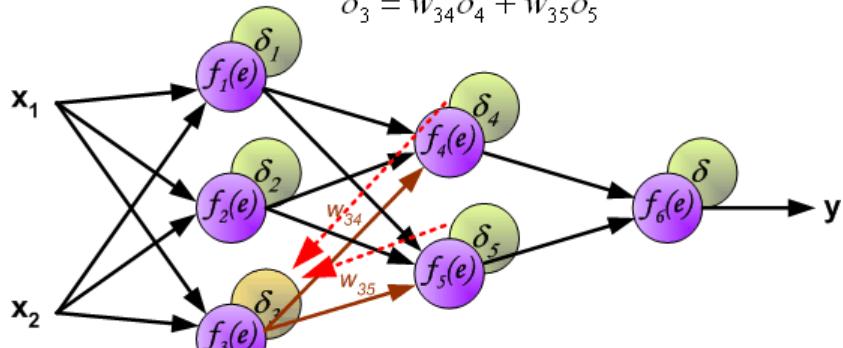
Пример



$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$



$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$

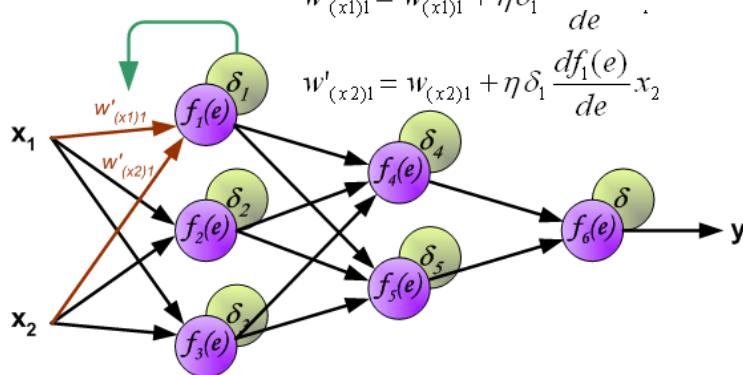


$$S(x) = \frac{1}{1 + \exp(-x)}$$

$$S'(x) = S(x) * (1 - S(x))$$

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1}{de}$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1}{de} x_2$$



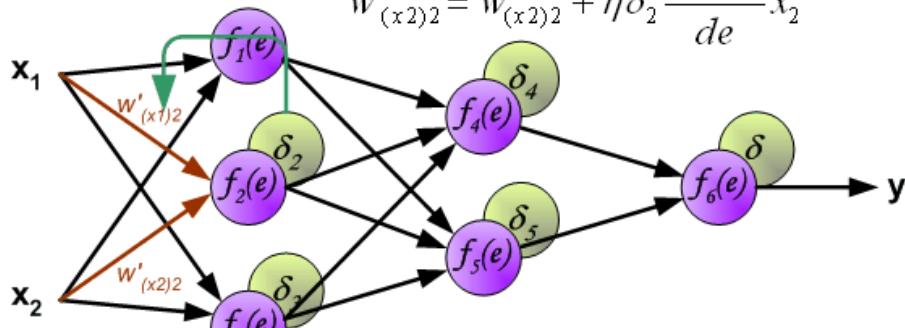
Нейронные сети

Метод обратного распространения ошибок

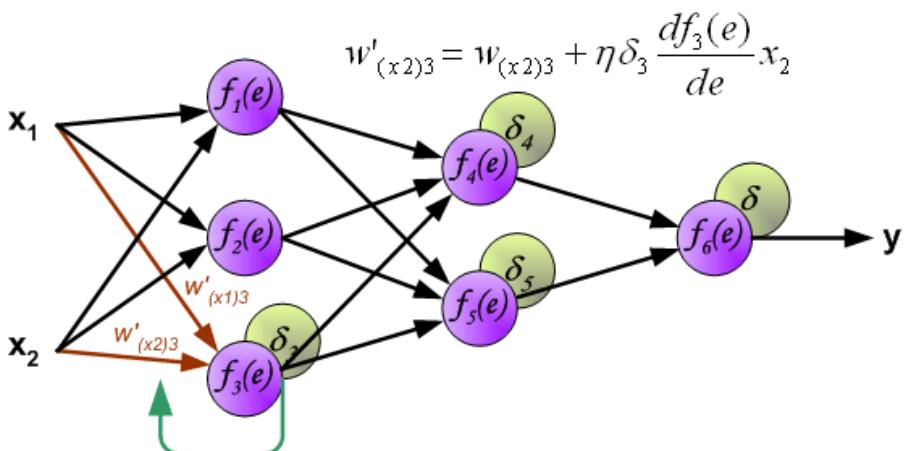
Пример

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

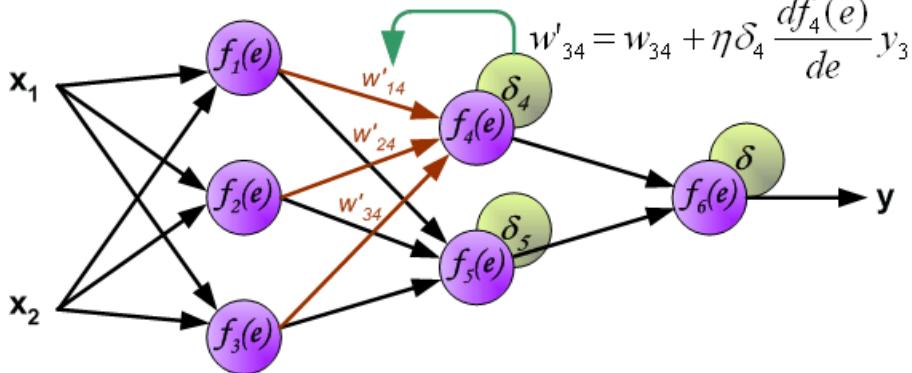


$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$



$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$



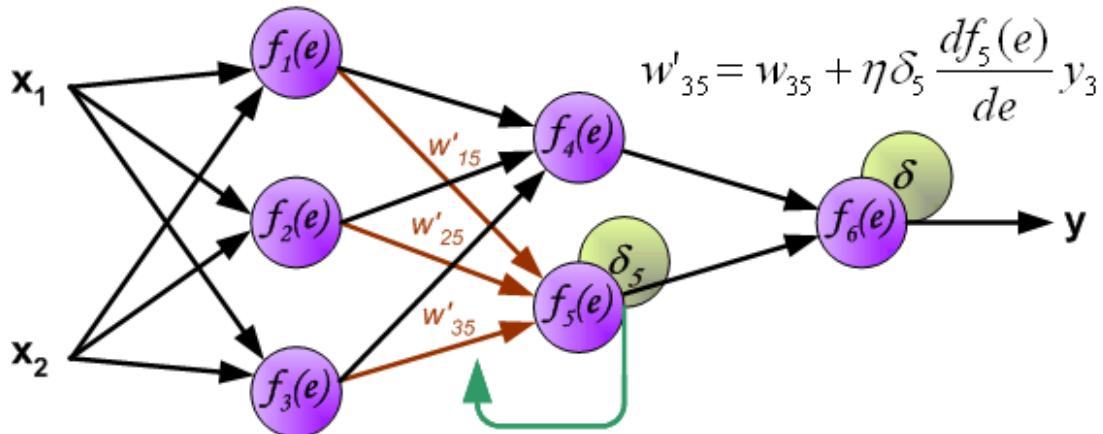
Нейронные сети

Метод обратного распространения ошибок

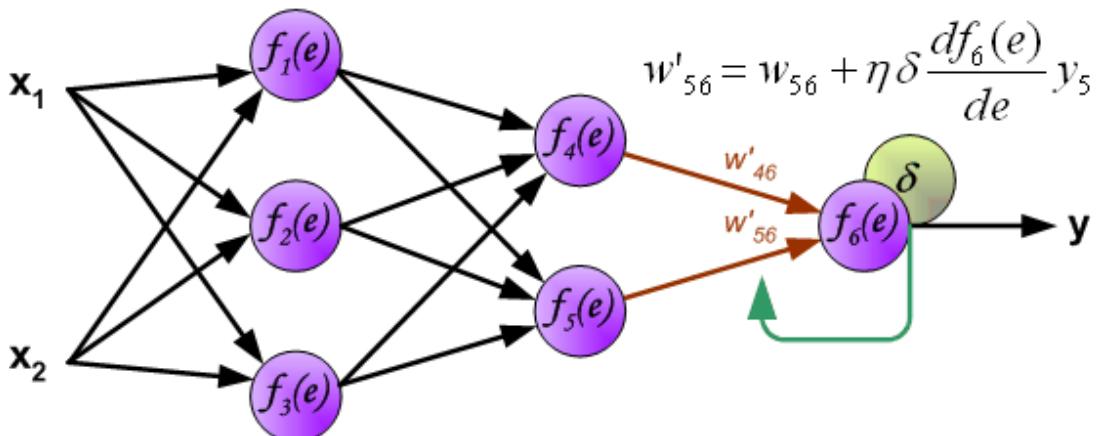
Пример

$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$



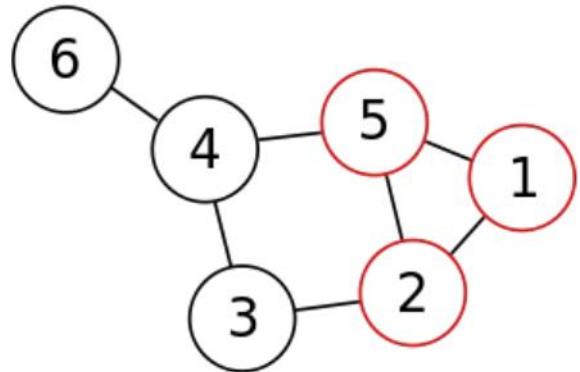
$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$



Клика

Определение

- Подмножество вершин графа, такое что любые две вершины подмножества соединены ребром.
- Подмножество вершин $C \subseteq V$ графа $G = (V, E)$, такое что для любой вершины из C найдется ребро их соединяющее.



Максимальная клика

Клика, которая не может быть расширена путем добавления дополнительных смежных вершин.

Наибольшая клика

Клика максимального размера для графа

Кликовое число

Число вершин в максимальной клике графа

Задача о клике

NP-задача

Задача распознавания

Есть ли в заданном графе G клика размера k ?

Задача поиска

Найти в G максимального размера.

Перебор

Рассматриваются все подграфы размера k графа G . Сложность - $\binom{v}{k} = \frac{v!}{k!(v-k)!}$.

Алгоритм Брана-Кербоша

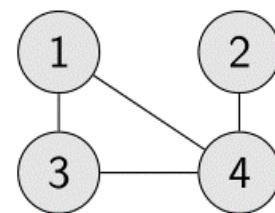
Сложность линейна относительно числа клик в графе.

Алгоритм Брана-Кербоша

- Находит все максимальные по включению независимые множества.
- На каждом шаге алгоритма множество V разбито на четыре части:
- M — текущее независимое множество;
- $\Gamma(M)$ — множество вершин, смежных с M ;
- K — множество кандидатов, т. е. вершин, каждая из которых может быть добавлена в M ;
- P — множество просмотренных вершин, каждая из которых не может быть добавлена в текущее M , так как уже добавлялась ранее.

Пример

M	K	P	v
\emptyset	1, 2, 3, 4	\emptyset	1
1	2	\emptyset	2
► 1, 2	\emptyset	\emptyset	
1	\emptyset	2	
\emptyset	2, 3, 4	1	2
2	3	1	3
► 2, 3	\emptyset	\emptyset	
2	\emptyset	1, 3	
\emptyset	3, 4	1, 2	3
3	\emptyset	2	
→ \emptyset	4	1, 2, 3	4
► 4	\emptyset	\emptyset	
\emptyset	\emptyset	1, 2, 3, 4	



while $K \neq \emptyset$ or $M \neq \emptyset$:

if $K \neq \emptyset$:

$v = K.\text{first}$

push M, K, P, v

$M = M + \{v\}$

$K = K - \Gamma(v) - \{v\}$

$P = P - \Gamma(v)$

else:

if $P == \emptyset$: вывод M

pop v, P, K, M

$K = K - \{v\}$

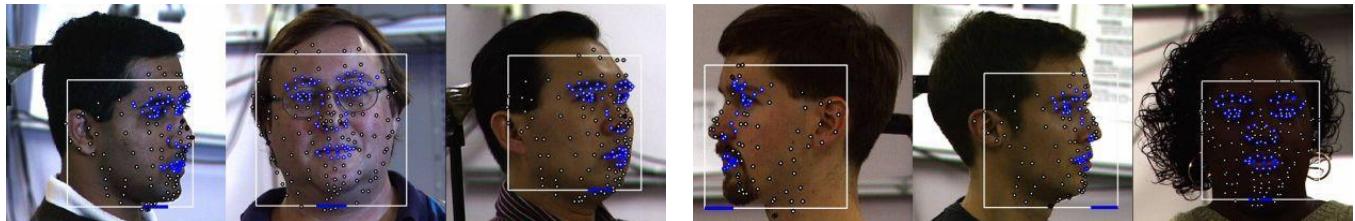
$P = P + \{v\}$

Вывод: {1,2} , {2,3} , {4}

Корреляционные методы анализа изображений.

Предобработка изображений

- Задача: получение инвариантного к условиям съемки входного изображения
- Основные элементы: точки, линии, границы

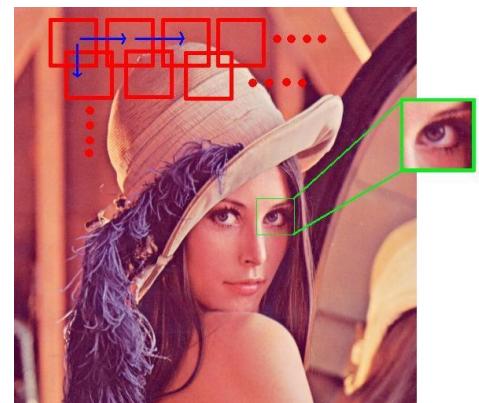


Корреляционный метод

- Выбирается шаблон – объект поиска
- Двигая шаблон по изображению, вычисляется матрица корреляций (на основе интенсивностей пикселей)
- Т.е. пытаемся «применить» шаблон к разным частям изображения
- $T(x_t, y_t)$ – матрица шаблона, где (x_t, y_t) – координаты пикселей
- $S(x, y)$ – матрица изображения, в котором ищем шаблон

$$SAD(x, y) = \sum_{i=0}^{T_{rows}} \sum_{j=0}^{T_{cols}} Diff(x + i, y + i, i, j)$$

- $Diff$ – модуль разности интенсивности пикселей $S(x + i, y + j)$ и $T(i, j)$



Достижение инвариантности

- Шаблон представляют во всех вариациях и запускают алгоритм для каждой
- Часто достаточно инвариантности к вращению, масштабированию и сдвигу

Вычислительная сложность

- Растет экспоненциально от количества степеней свободы шаблона
- Уменьшение сложности: построение пирамид изображения (уменьшение масштаба, прогонка и наращивание масштаба изображения для самых перспективных областей)

Корреляционные методы анализа изображений.

Метод быстрой корреляции

- Предварительные действия: подготовка набора шаблонов
- Определение области, в которой будет производиться сравнение с шаблонами (вручную или автоматически)
- Формирование корреляционных полей (прогонка шаблонов по области)
- Обработка корреляционных полей (отбор перспективных точек)
- Принятие решения (анализ корреляционных полей)

Подготовка шаблонов

- Должны максимально точно соответствовать искомым объектам
- Минимизация влияния мешающих факторов и снижение вычислительной мощности
- Тернарные шаблоны (отражают область объекта и фона) – свертка изображения с шаблоном без умножения, суммированием

Алгоритм распознавания (символов)

- Определяется положение очередного символа
- Путем корреляционного сравнения изображения символа с множеством шаблонов определяются наиболее похожие
- Шаги повторяются для всех символов строки; в результате формируется матрица решений
- Выбираются варианты, удовлетворяющие всем семантическим ограничениям, и формируется распознанная строка

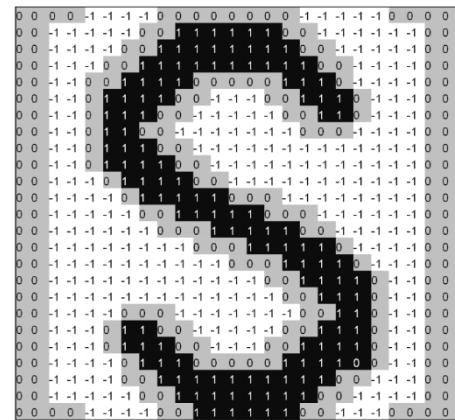


Рис.1. Шаблон символа S (серым цветом – неучитываемые элементы, белым – фон, темным – символ)

Особенности

- Используются тернарные шаблоны с учетом специфики шрифтов
- Вероятность верного распознавания символа – 0,998, всей строки – 0,982



Рис.3. Шаблоны символов «цифра»

Склейка изображений.

Применение

- Фотографии с высоким разрешением, получаемые из склейки нескольких изображений
- Захват сцен, которые нельзя получить за один кадр
- Дешёвый и простой способ достичь того, что стоило бы больших денег
- Как HDR и Focus Stacking, использует численные методы, чтобы выйти за рамки возможностей камеры

Методика захвата изображений для панорамы

- Необходимо наложение ($>=15\%$, чтобы уменьшить эффект дисторсии)
- Выдержка должна быть примерно одинаковой для более плавных переходов
- Нужно относиться с осторожностью к: дисторсии, поляризующим фильтрам, резкости в области наложения

Обзор алгоритма

- Найти ключевые точки (например, с помощью SURF)
- Совместить ключевые точки по характерным признакам
- Вычислить матрицу проективного преобразования по четырём совпавшим точкам, используя RANSAC
- Проектировать на поверхность и наложить друг на друга
- Сгладить получившиеся швы

Обзор RANSAC

- Повторить N раз (N выбираем сами)
- Случайно выбрать набор пар ключевых точек: выбрать достаточно точек, чтобы восстановить параметры
- Подставить значения для подсчёта матрицы проективного преобразования
- Посчитать количество других пар с таким же значением матрицы
- Итоговое значение матрицы – значение, к которому пришло наибольшее число пар

Склейка изображений.

Alpha Blending

- Смешивание двух изображений в области наложения производится усреднённой суммой значений пикселей с весом
- Веса хранятся в маске. Чёрные пиксели соответствуют пикселям первого изображения, белые – второго. Промежуточные значения доставляют смешанные пиксели по формуле:

$$pix_{result} = \alpha * pix_2 + (1 - \alpha) * pix_1$$

Вычисление маски

- Вычисление общей области исходя из решения, предложенного SURF
- Обнаружение границ общей области
- Для каждой точки общей области вычисляется расстояние до изображения 1 и изображения 2.
- Значение вычисляется по формуле:



$$\begin{cases} 1 - \frac{d_2}{2d_1} & \text{if } d_1 \geq d_2 \\ \frac{d_1}{2d_2} & \text{if } d_1 < d_2 \end{cases}$$



Работа с большими изображениями

- Если изображение достаточно большое (больше 1000*1000 пикселей), то это сильно замедляет работу алгоритма и требует больших ресурсов.
- Эта проблема решается тем, что изображения сначала уменьшаются в несколько раз, по ним высчитывается маска, которая впоследствии растягивается до размеров исходного изображения.

