§30. Лабораторная работа № 6
Кластеризация данных

Ц е л ь р а б о т ы: Получение навыков практического применения иерархических и итеративных методов кластерного анализа.

С о д е р ж а н и е р а б о т ы: реализовать алгоритмы кластерного анализа.

Методические указания

Название кластерный анализ происходит от английского слова cluster – гроздь, скопление. Кластерный анализ – широкий класс процедур многомерного статистического анализа, позволяющих произвести автоматизированную группировку наблюдений в однородные классы – кластеры.

Кластер имеет следующие математические характеристики:

- центр;
- радиус;
- дисперсия кластера;
- среднеквадратическое отклонение.

Центр кластера – это среднее геометрическое место точек в пространстве переменных.

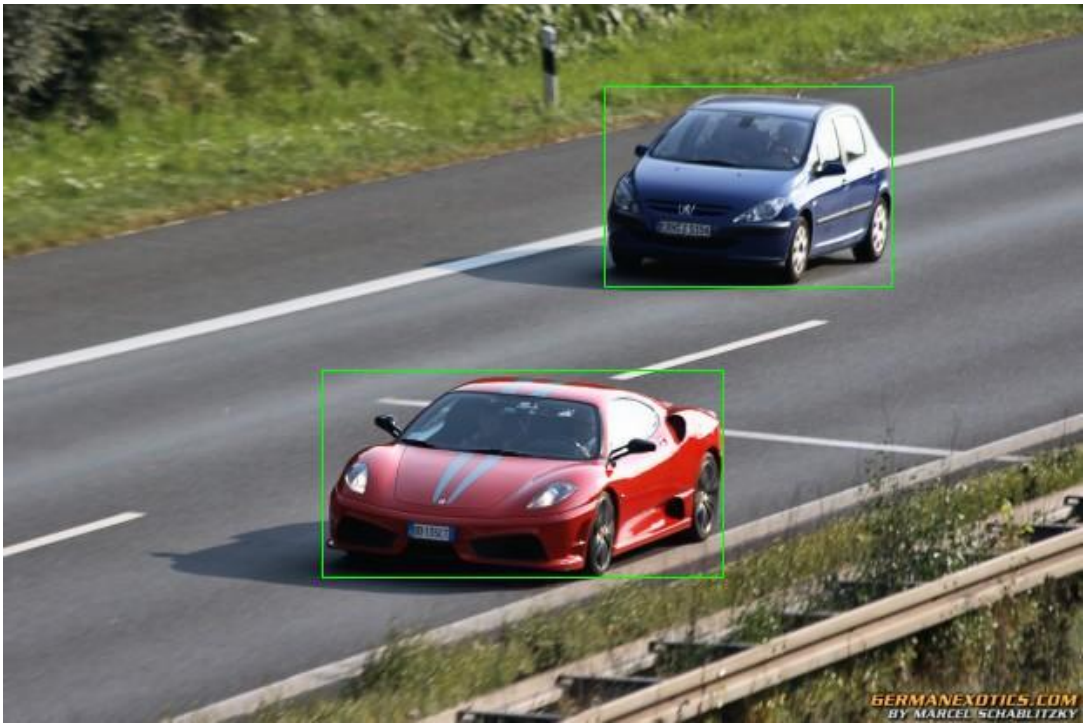
Радиус кластера – максимальное расстояние точек от центра кластера.

Дисперсия кластера – это мера рассеяния точек в пространстве относительно центра кластера.

Среднеквадратичное отклонение (СКО) объектов относительно центра кластера – квадратный корень из дисперсии кластера.

Простейшая кластеризация методом k-средних (k-means)

Зачастую при поиске движущихся объектов на видео будь то методом вычитания фона, временной разности, оптического потока, в итоге мы получаем множество точек, которые после действия вышеупомянутых алгоритмов помечены как изменившие свое положение относительно предыдущего кадра и относящиеся к переднему плану.



После такой обработки встает вопрос о сегментации объектов методом кластерного анализа, о котором пойдет речь ниже и собственно его реализация на *C++*.

Сегментация объектов

Для начала немного теории. Сегментация – это процесс разделения цифрового изображения на несколько сегментов (множеств пикселей). Проще говоря, это вещь, которая позволяет определить какие пиксели из данного множества относятся к Ferrari, а какие к Peugeot. Очень эффективным с точки зрения вычислительных ресурсов является использование для сегментации методов кластерного анализа. Суть кластеризации состоит в том, что все исходные объекты (в данном случае пиксели) разбиваются на несколько не пересекающихся групп таким образом, чтобы объекты, попавшие в одну группу, имели сходные характеристики, в то время как у объектов из разных групп эти характеристики должны значительно отличаться. Полученные группы называются кластерами. Исходными значениями в простейшем способе для кластеризации являются координаты пикселя (x, y), в более сложных случаях, например для полутоновых изображений, используется трехмерный вектор (x, y, I(x, y)), где I(x, y) – градации серого, пятимерный вектор если используется RGB.

Метод k-средних

Центроид – точка которая является центром кластера. *k*-средних (*k*-means) — наиболее популярный метод кластеризации. Алгоритму широко отдается

предпочтение из-за его простоты реализации, большой скорости (а это очень важно при работе с видео). Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. В простонародье говоря, это итеративный алгоритм, который делит данное множество пикселей на k кластеров точки, которых являются максимально приближенными к их центрам, а сама кластеризация происходит за счет смещения этих же центров. Такой себе принцип разделяй и властвуй. Также следует оговорить то, что метод к-средних очень чувствительный к шуму, который может существенно исказить результаты кластеризации. Так что в идеале, перед кластеризацией, нужно прогнать кадры через фильтры предназначенные для его уменьшения. Вот собственно сам принцип простейшей кластеризации методом *k* -средних:

1. Надо выбрать из множества k пикселей те пиксели, которые будут центроидами соответствующих *k* кластеров.

2. Выборка начальных центроидов может быть, как рандомной так и по определенному алгоритму.

3. Входим в цикл, который продолжается до тех пор, пока центроиды кластеров не перестанут изменять свое положение.

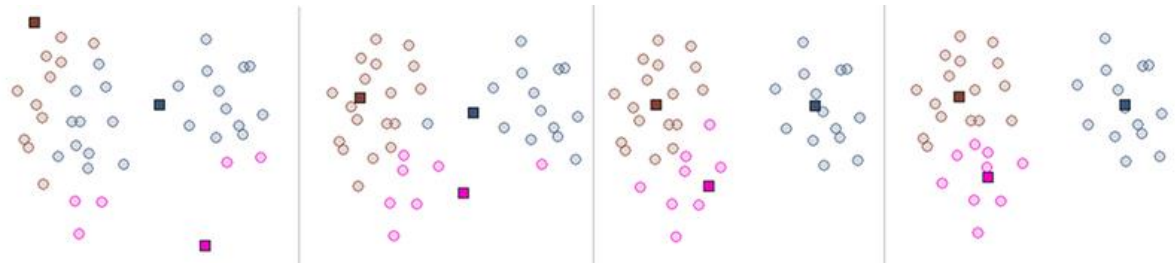
4. Обходим каждый пиксель и смотрим, к какому центроиду какого кластера он является близлежащим.

5. Нашли близлежащий центроид? Привязываем пиксель к кластеру этого центроида.

6. Перебрали все пиксели? Теперь нужно высчитать новые координаты центроидов *k* кластеров.

7. Теперь проверяем координаты новых центроидов. Если они соответственно равны предыдущим центроидам – выходим из цикла, если нет возвращаемся к пункту 3.

Вот рисунок, который приблизительно демонстрируют работу алгоритма:



Для начала нам нужен класс, назовем его Cluster, который будет хранить вектор координат пикселей, относящихся к кластеру, текущие и предыдущие значения координат центроида:

```
lass Cluster{
	vector<POINT> scores;
public:
	int curX , curY;//координаты текущего центроида
	int lastX, lastY;//координаты предыдущего центоида
	size_t Size(){ return scores.size();}//получаем размер вектора
	inline void Add(POINT pt){ scores.push_back(pt); }//Добавляем пиксель к кластеру
	void SetCenter();
	void Clear();//Чистим вектор
	static Cluster* Bind(int k, Cluster * clusarr, vector<POINT>& vpt);
	static void InitialCenter(int k, Cluster * clusarr , vector<POINT>& vpt);;
	static void Start(int k, Cluster * clusarr, vector<POINT>& vpt);
	inline POINT& at(unsigned i){ return scores.at(i);}//Доступ  к элементам вектора
};
```

Теперь нам надо реализовать метод которой будет распределять начальные координаты центроидов. Можно конечно сделать чего-нибудь по сложнее, но в нашем случае сойдет и равномерное распределение по вектору:

```
void Cluster::InitialCenter(int k, Cluster * clusarr, vector<POINT>& vpt){

	int size = vpt.size();
	int step = size/k;
	int steper = 0;

	for(int i = 0;i < k;i++,steper+=step){
		clusarr[i].curX = vpt[steper].x;
		clusarr[i].curY = vpt[steper].y;
	}
}
```

Также нужно написать метод, который будет ответственный за нахождение новых координат центроида в соответствии с пунктом 5.Координаты нового центроида можно найти описав вокруг пикселей кластера прямоугольник и тогда центроидом будет пересечение его диагоналей.

```
void Cluster::SetCenter(){
	int sumX = 0, sumY = 0;
	int i = 0;
	int size = Size();
	for(; i<size;sumX+=scores[i].x,i++);//the centers of mass by x
	i = 0;
	for(; i<size;sumY+=scores[i].y, i++);//the centers of mass by y
	lastX = curX;
	lastY = curY;
	curX = sumX/size;
	curY = sumY/size;
}

void Cluster::Clear(){
	scores.clear();
}
```

И теперь только остался сделать простенький метод самого «привязывания» пикселей к определенному кластеру по принципу сравнения модулей отрезков:

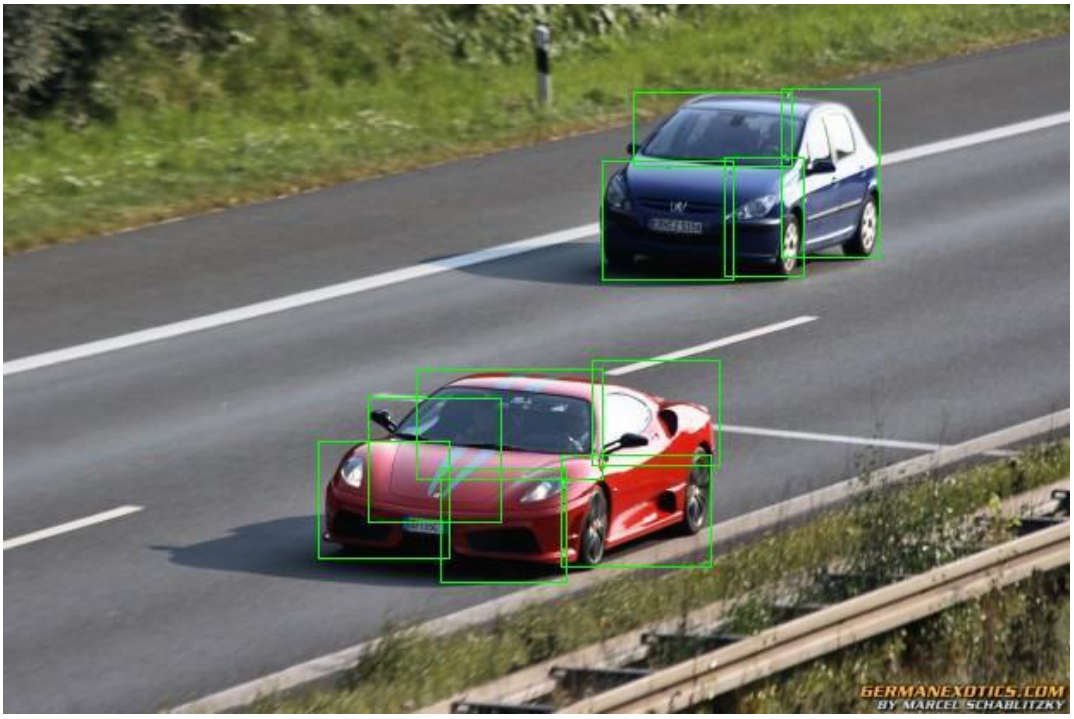
```
Cluster * Cluster::Bind(int k, Cluster * clusarr, vector<POINT>& vpt){
	for(int j = 0; j < k;j++)
		clusarr[j].Clear();// Чистим кластер перед использованием
	int size = vpt.size();
		for(int i = 0; i < size; i++){// Запускаем цикл по всем пикселям множества
			int min = sqrt(
				pow((float)clusarr[0].curX-vpt[i].x,2)+pow((float)clusarr[0].curY-
vpt[i].y,2)
				);
			Cluster * cl = &clusarr[0];
			for(int j = 1; j < k; j++){
				int tmp = sqrt(
					pow((float)clusarr[j].curX-
vpt[i].x,2)+pow((float)clusarr[j].curY-vpt[i].y,2)
					);
				if(min > tmp){ min = tmp; cl = &clusarr[j];}// Ищем
близлежащий кластер
```

```
                }
                cl->Add(vpt[i]);// Добавляем в близ лежащий кластер текущий
пиксель
        }
    return clusarr;
}
```

И наконец главный цикл:

```
void Cluster::Start(int k, Cluster * clusarr, vector<POINT>& vpt){
    Cluster::InitialCenter(k,clusarr,vpt);
    for(;;){//Запускаем основной цикл
        int chk = 0;
        Cluster::Bind(k,clusarr,vpt);//Связываем точки с кластерами
        for(int j = 0; j < k;j++)//Высчитываем новые координаты центроидов
            clusarr[j].SetCenter();
        for(int p = 0; p<k;p++)//Проверяем не совпадают ли они с предыдущими
цент-ми
            if(clusarr[p].curX == clusarr[p].lastX && clusarr[p].curY ==
clusarr[p].lastY)
                chk++;
        if(chk == k) return;//Если да выходим с цикла
    }
}
```

Вывод. Вернемся к картинке с машинами, кластеризуя движущиеся объекты возникает проблема при использовании алгоритма к-средних, а именно мы не знаем сколько в данной сцене будет движущихся объектов, хотя можем приблизительно предугадать. Например, кадр с машинами, на той сцене разумным будет предположить, что ну максимум там будет машин 10. Таким образом задавая на вход программе $k = 10$ и обведя точки 10 кластеров зелеными прямоугольниками, мы получим примерно следующую картину:



Теперь банально объединив пересекающиеся прямоугольники, мы находим результирующие кластеры, обведя которые прямоугольником мы получим изображение, приведённое в начале.

Варианты заданий к лабораторной работе № 9

Варианты № 1 – № 5.

Пусть $x=(x_1,...,x_9)$ – случайная 9-мерная величина, координаты которой распределены равномерно в интервале (3,8). Произвести 50 ее реализаций и распределить их на минимальное число кластеров, каждый из которых помещается в сфере радиуса 0.15.

Варианты № 6 – № 10.

Имеется 5-мерная случайная величина $x=(x_1,...,x_5)$, координаты которой распределены по показательному закону с функцией распределения $f(t)=1-exp(-2t)$, $t \geq 0$. Фиксируется 50 реализаций этой случайной величины. Требуется распределить эти реализации на минимальное число кластеров, внутри которых дисперсия каждой координаты не превосходит 0.25.

Варианты № 11 – № 15.

Имеется 6-мерная случайная величина $x=(x_1,...,x_6)$, координаты которой распределены нормально с дисперсией 1 и математическим ожиданием 0. Фиксируется 50 реализаций этой случайной величины. Требуется распределить эти

реализации на минимальное число кластеров так, чтобы математические ожидания $m_1,...,m_6$ координат в одном кластере составляли точку, удаленную от любой точки кластера меньше, чем на 0.3.

Варианты № 16 – № 20.

Пусть $x=(x_1,...,x_7)$ – семимерная случайная величина, координаты которой являются нормально распределенными случайными величинами с математическим ожиданием 0.1 и дисперсией 0.2. Фиксируется 50 реализаций величины x. Требуется распределить это множество реализаций на минимальное число кластеров так, чтобы внутри каждого кластера изменения координат x_1, x_3, x_5, x_7 изменялись не больше, чем на 0.5.

Варианты № 21 – № 25.

Пусть $x=(x_1,x_2,x_3)$ – трехмерная случайная величина, координаты которой являются экспоненциально распределенными случайными величинами с плотностью распределения $f(t)=1-exp(-3.5t)$. Фиксируется 50 реализаций величины x. Требуется распределить эти реализации на минимальное число кластеров так чтобы каждый из них помещался в кубе, ребра которого параллельны осям координат и имеют длину 0.3.

Варианты № 26 – № 30.

Пусть $x=(x_1,...,x_7)$ – семимерная случайная величина, координаты которой являются нормально распределенными случайными величинами с математическим ожиданием 0.1 и дисперсией 0.2. Фиксируется 50 реализаций величины x. Требуется распределить это множество реализаций на минимальное число кластеров так, чтобы внутри каждого кластера изменения координат x_1, x_3, x_5, x_7 изменялись не больше, чем на 0.5.