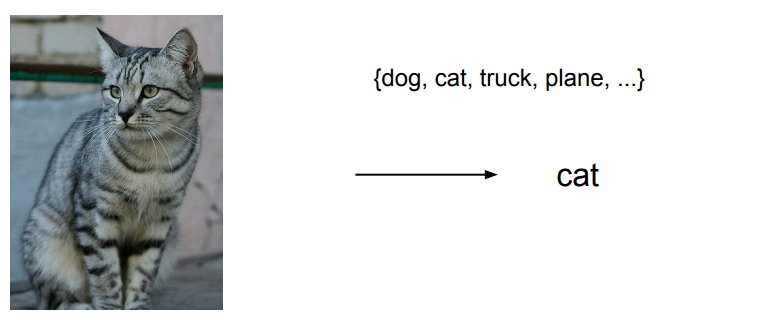
**Лекция №6. Классификация изображений.**

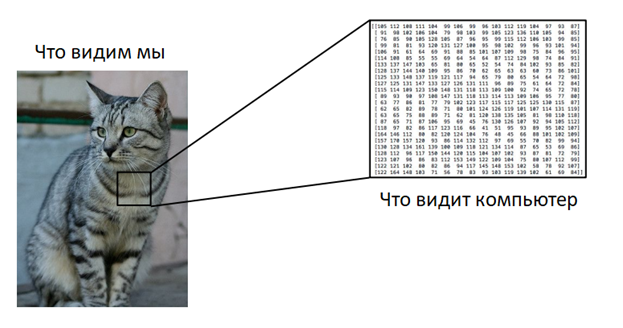
Когда ваша система получает входное изображение, ей уже известен фиксированный набор категорий или меток. Это могут быть любые объекты: «кошка», «собака», «самолёт», «грузовик» и так далее. Компьютер должен посмотреть на изображение и назначить ему одну из меток.



Со стороны задача выглядит несложной, поскольку большая часть нашей зрительной системы запрограммирована на распознавание объектов. Но для машины это не так-то просто.

**Проблема классификации**

Когда компьютер смотрит на изображение, то не видит целостное представление кошки или любого другого объекта. Он видит лишь гигантскую сетку чисел. Например, если размер изображения 800 на 600 и каждый пиксель представлен тремя числами для красного, зелёного и синего каналов, то получится сетка из 800х600х3 = 1,440,000 чисел. Очень трудно выделить из них какой-то конкретный предмет, представленный на фотографии.



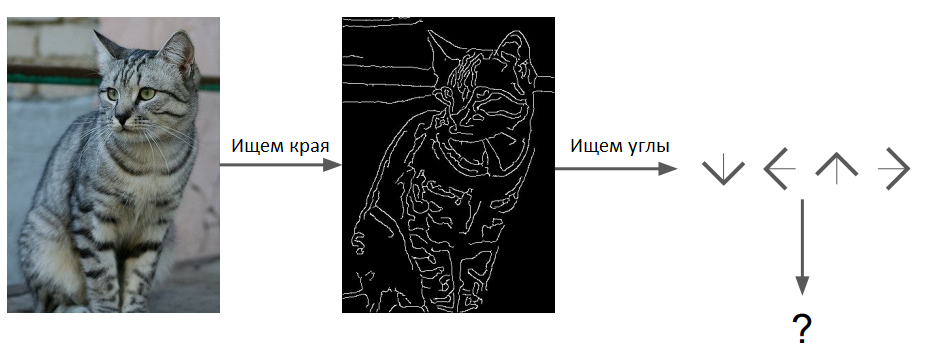
Эта проблема получила название «семантический разрыв» – непонимание информации, которая заключена в данных. Например, если мы снимем кошку с другого ракурса или при ином освещении, то вся сетка чисел будет выглядеть совершенно иначе. Помимо этого, животные могут принимать множество различных поз, или же на фотографии может оказаться только часть кошки, например, хвост. Алгоритмы распознавания должны быть устойчивы к таким изменениям.



Помимо этих сложностей существует ещё проблема внутриклассовых вариаций, когда одно понятие охватывает множество визуальных проявлений. Например, кошки могут быть разных пород, возрастов и размеров. И методы распознавания должны обрабатывать все возможные варианты.

**Пытаемся решить проблему**

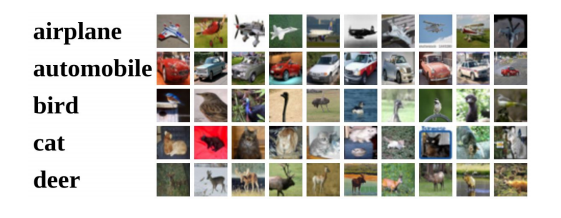
Первое, что приходит в голову – создать эталонные правила. Мы знаем, что у кошек есть уши, глаза, рот и нос. На фотографиях с ними мы можем обнаружить края, а затем классифицировать разные углы и границы: например, определить, каким образом пересекаются линии ушей и носа.



Но явные наборы правил работают не очень хорошо: любое отклонение может всё сломать, а для новых объектов придётся создавать новые условия. Поэтому такой подход не является масштабируемым.

**Подход, основанный на данных**

Вместо того, чтобы вручную пытаться создать набор правил, можно открыть интернет и собрать большой набор данных с фотографиями кошек, самолётов, оленей и других вещей. Для этого подойдёт поиск картинок Google или уже готовый датасет. Затем необходимо обучить классификатор, отправив в него все собранные изображения. На выходе мы получим модель, обобщающую знания о распознавании разных объектов. После этого она сможет работать на новых изображениях и отличать кошек от собак.

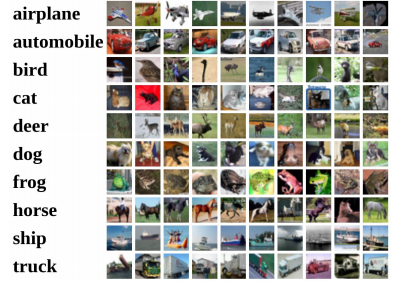


Итак, вместо одной функции, которая просто распознаёт объект на входном изображении, мы получили две: первая называется «обучение» – это процесс обработки изображений и создания модели. Вторая функция – «прогноз» – распознаёт новые фотографии. Вместе они составляют основу для свёрточных нейронных сетей и глубокого обучения в целом.

**Метод ближайшего соседа**

Для начала рассмотрим самый простой классификатор, который носит название «метод ближайшего соседа». В процессе обучения он запоминает все исходные данные, а затем на этапе прогнозирования пытается найти среди них наиболее похожие на новое изображение.

Возьмём набор данных CIFAR-10, содержащий 10 различных классов и около 50000 обучающих и 10000 тестовых образцов.



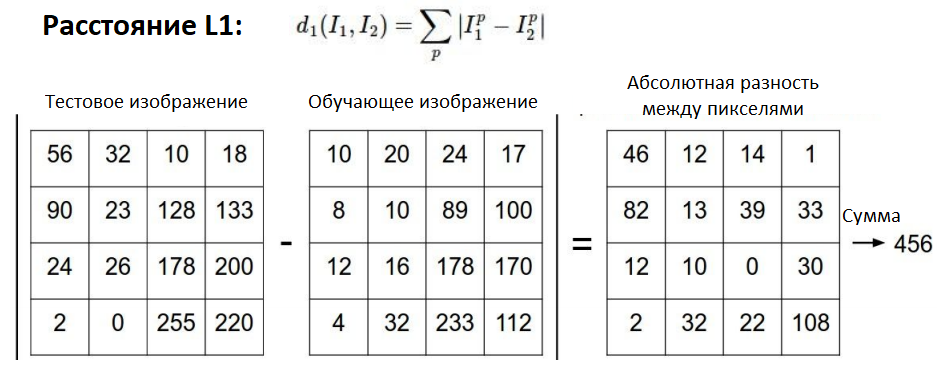
Так на нём работает классификатор ближайшего соседа:



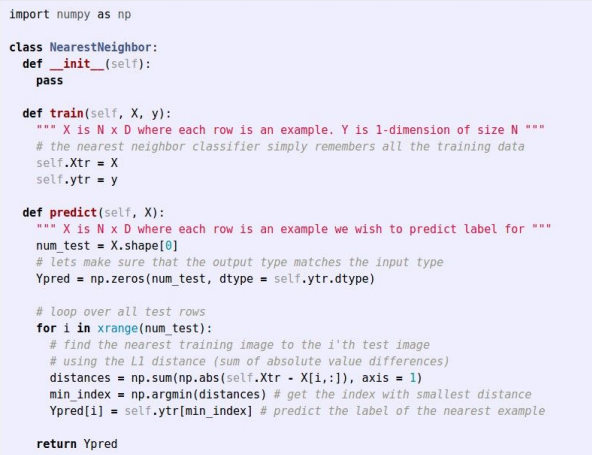
Слева показаны тестовые изображения, а справа – наиболее похожие на них обучающие образцы. Можно заметить, что они не всегда оказываются правильными. Взяв наиболее близкого соседа и его метку, алгоритм классифицирует тестовое изображение.

**Как это работает?**

Как же сравнить два изображения? На самом деле для этого есть много разных способов. В примере выше использовалось манхэттенское расстояние. Оно просто сравнивает между собой пиксели фотографий. Предположим, что у нас есть тестовый образец размером 4х4 пикселя. Возьмём одно из обучающих изображений и вычислим абсолютную разницу между цветами пикселей обучающего и тестового образцов, а затем просуммируем полученные значения.



Что именно делает алгоритм ближайшего соседа? На рисунке ниже показаны так называемые области принятия решений классификатора.

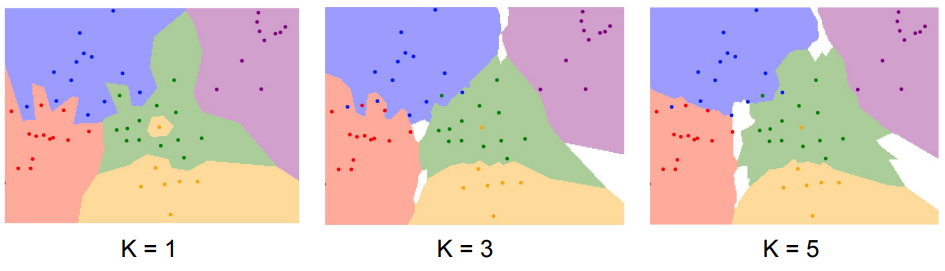


Точки – это объекты обучающей выборки, а цвета – категории или метки классов. Для каждого образца определяется наиболее похожий пример, и в соответствии с ним точка закрашивается цветом найденного класса.

Но данные могут быть зашумлёнными или некорректными. Для решения этой проблемы существует более общий метод.

**Метод k-ближайших соседей**

Вместо того, чтобы искать одно похожее изображение, мы будем искать k ближайших соседей в соответствии с метрикой расстояния. После этого проведём голосование по каждому соседу и спрогнозируем метку согласно большинству голосов. Здесь показан тот же набор точек с числом соседей k=1, k=3 и k=5.



Можно увидеть, что жёлтая точка внутри зелёного кластера больше не создаёт область вокруг себя, а граница между красным и синим классами становится более гладкой. Как правило, для классификации используется значение k > 1. Белые регионы здесь означают области, в которых не найдено ни одного ближайшего соседа.

*Вы можете самостоятельно протестировать классификатор ближайшего соседа в*[*веб-демонстрации*](http://vision.stanford.edu/teaching/cs231n-demos/knn/)*. Попробуйте поиграть с метриками, числом соседей, классов и точек.*

**Гиперпараметры**

Процесс выбора влияющих на работу какого-либо метода значений, таких как число соседей k и метрика расстояния, называется настройкой гиперпараметров. Гиперпараметры не могут быть явно извлечены из обучающих данных и зависят только от самого алгоритма, поэтому не существует чётких рекомендаций по их выбору. Чаще всего приходится искать значения методом проб и ошибок, выясняя, какие работают лучше всего.

Разберёмся, что такое «лучше всего».

**Идея 1.**Подобрать такие параметры, которые дадут самую высокую точность на обучающих данных.

И это **очень** **плохая** идея. Никогда не делайте так. В случае с методом ближайших соседей при k=1 достигается почти идеальная точность во время обучения, но с тестовыми данными алгоритм справляется крайне плохо. Это явление называется «переобучение».

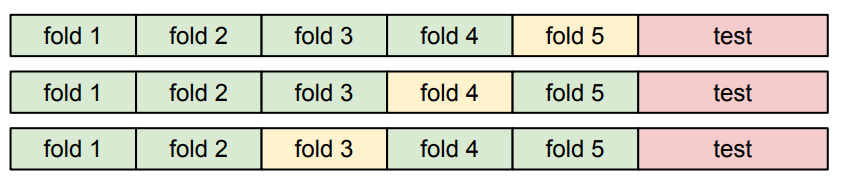
**Идея 2.** Разделить набор на обучающую и тестовую выборки и найти гиперпараметры, которые заставят алгоритм лучше работать на тестовых образцах.

Эта стратегия выглядит более разумной, но на самом деле она тоже **очень плоха**. Основная идея машинного обучения в том, что мы хотим знать, как будет работать метод. И если мы подбираем параметры, с которыми достигаются хорошие результаты на неизвестных изображениях, то нет никакой гарантии, что они будут достигнуты на других неизвестных изображениях.

**Идея 3.**Разделить набор данных на обучающую, тестовую и оценочную выборки; подобрать гиперпараметры для оценочных данных и проверить на тестовых.

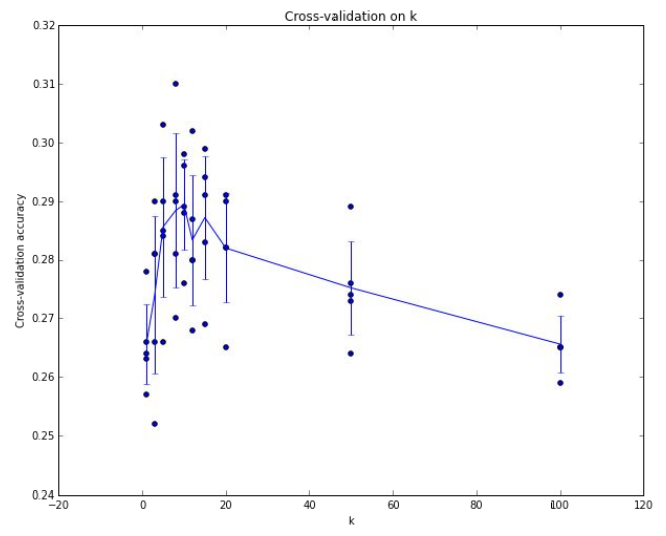
И это уже **хорошая** идея. Сначала классификатор обучается с различными вариантами параметров. Затем выбираются те значения, которые лучше всего работают на оценочных данных. После этого модель обрабатывает тестовую выборку только один раз. Точность, которая достигается при этом, показывает истинную эффективность классификатора.

**Идея 4.** Разделить обучающие данные на множество небольших подвыборок, затем сделать перекрёстную проверку, используя разные подвыборки в качестве оценочных данных, и усреднить результаты.



Метод называется «кросс-валидация» и хорошо работает на небольших датасетах, но требует очень много вычислительных ресурсов для огромных наборов данных. По этой причине такой подход нечасто используется в глубоком обучении.

После перекрёстной проверки вы получите график, который выглядит примерно так:



На оси X показано значение k (число соседей), а на оси Y – точность классификатора. По графику можно оценить эффективность модели в зависимости от гиперпараметров. В данном случае наибольшая точность достигается при k≈7.

**Теперь я могу классифицировать изображения с котиками?**

К сожалению, нет. На самом деле метод k-ближайших соседей никогда не применяется к фотографиям, потому что:

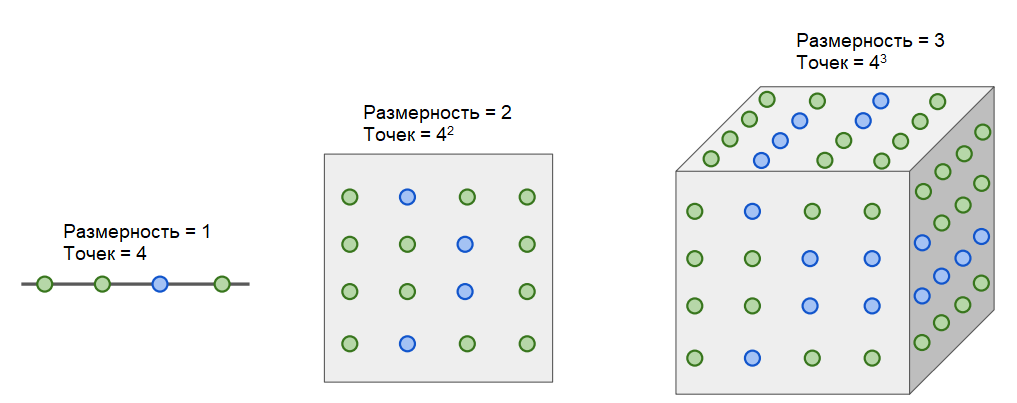
– очень медленно работает с тестовыми данными

– метрика расстояний между цветами пикселей не говорит о сходстве изображений.



В этом примере к исходному снимку (слева) применяются различные искажения: закрашивание глаз и рта, сдвиг на несколько пикселей и тонирование синим цветом. Если мы вычислим евклидово расстояние между оригиналом и любой из изменённых фотографий, то во всех случаях оно будет одинаковым. И это не очень хорошо, ведь метрика L2 должна отражать разницу между изображениями.

Ещё одна проблема – так называемое «проклятие размерности», связанное с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.



Пример выше показывает, что в одномерном случае четырёх обучающих образцов достаточно, чтобы покрыть пространство с необходимой плотностью. В двумерном измерении для той же степени покрытия потребуется уже 16 точек, а в трёхмерном – 64. Поэтому алгоритмы, основанные на переборе, становятся неэффективными при возрастании размерности системы.

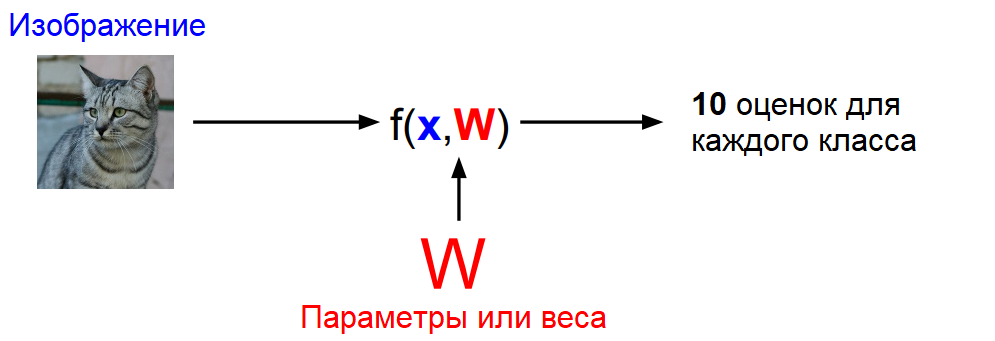
**Линейная классификация**

Алгоритмы линейной классификации довольно просты. Тем не менее, с их помощью создаются полноценные нейронные сети. Это похоже на сборку Lego: можно сложить друг с другом различные компоненты и построить «башню» свёрточной нейросети.



Линейный классификатор – один из основных строительных блоков, использующихся в различных приложениях глубокого обучения. Попробуем разобраться, как он работает.

Вернёмся к набору данных CIFAR-10. Чтобы классифицировать изображения, мы создадим простую параметрическую модель из двух компонент. Первая – входные данные, обычно обозначающиеся как **X**, и вторая – набор параметров или весов **W**. Теперь запишем некоторую функцию, которая принимает данные **X** и параметры **W**, а затем выдаёт 10 чисел, описывающих оценки для каждой из 10 категорий в CIFAR-10.

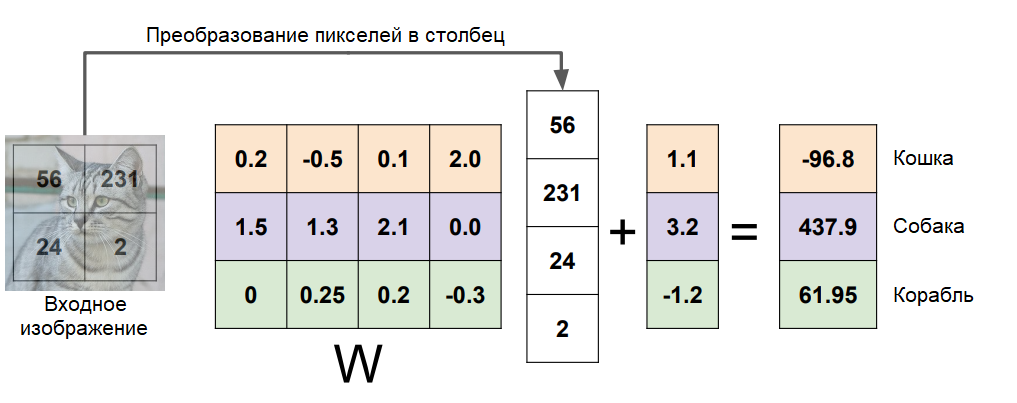


Правильно работающая модель в примере выше выдаст наибольшую оценку для класса кошки.

В этом параметрическом подходе, в отличие от метода ближайшего соседа, мы обобщаем знания об обучающих данных и используем их при создании параметров. Поэтому во время тестирования нам не нужно каждый раз обращаться к обучающей выборке.

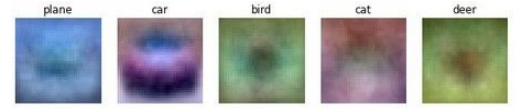
**Интерпретация линейного классификатора**

Рассмотрим простое изображение 2х2. Линейный классификатор растягивает его в вектор-столбец с четырьмя элементами. Здесь мы используем три класса: «кошка», «собака» и «корабль», поэтому весовая матрица будет иметь размерность 4х3. Вектор смещения из трёх элементов создаёт независимые условия для каждой категории:



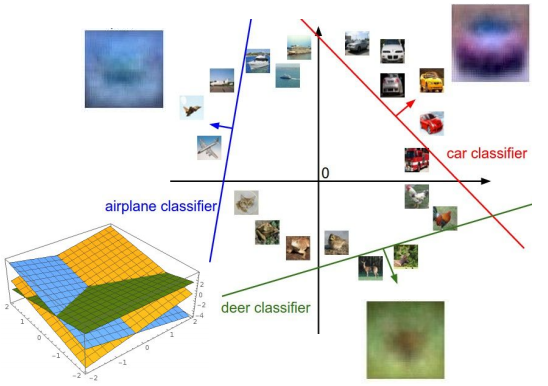
Оценка для любого из классов состоит из произведения между пикселями изображения и соответствующими строками весовой матрицы, к которому добавляется смещение. Это похоже на сравнение с образцом: каждая строка соответствует некоторому шаблонному изображению, а оценка указывает на его сходство с исходной фотографией.

Если попробовать свернуть строки весовой матрицы обратно в изображения, мы действительно получим шаблоны, собранные из обучающих данных:



Линейный классификатор использует только один шаблон для каждого класса. Создавая более сложные функции, связывающие данные и параметры, мы сможем изучать больше шаблонов и достигнем гораздо лучшей точности.

В области принятия решений каждое из изображений представлено в виде точки в многомерном пространстве, которую линейный классификатор пытается поместить в границы линейного решения. Иначе говоря, он будет разделять категории между собой прямыми линиями:



Но такое разделение между классами возможно далеко не всегда.

**Проблемы**

Приведём примеры данных, на которых линейный классификатор потерпит неудачу.

Предположим, что у нас есть датасет с двумя классами, в тривиальном случае – «синий» и «красный». «Синий» означает нечётное количество пикселей на изображении, а «красный» – чётное. Если мы попробуем изобразить эти области принятия решений на плоскости, то получим следующую картину:

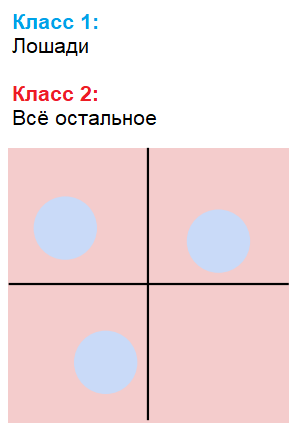


Видим, что синий и красный классы занимают противоположные квадранты, которые нельзя разделить одной прямой линией. Это называется проблемой паритета – когда два или более признаков равны по каким-либо параметрам.

Ещё одна непростая задача – явление мультимодальности, встречающееся в неравномерно распределённых данных. Например, фотографии животных могут изображать их с разных сторон, как в примере шаблона для класса «лошадь»:



При этом в области принятия решений появляются изолированные острова с лошадьми, смотрящими вправо, влево и в другие стороны, что выглядит примерно так:



Несмотря на возникающие проблемы, линейная классификация – простой алгоритм, который легко интерпретировать и реализовать.