Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
Кафедра компьютерных технологий

Курсовая работа по теме:

"Классификатор для автоматического распознавания рукописных цифр"

Выполнила:
Воробьева Валентина,
студентка группы 4539
Научный руководитель:
Антон Банных

2014

Описание предметной области

Распознавание рукописных цифр - важная задача во многих современных приложениях, таких как автоматическая сортировка почты по почтовому коду, автоматизированное чтение чеков или ввод данных для приложений современных гаджетов.

В этой области уже достигнут значительный прогресс, отчасти из-за применения различных алгоритмов машинного обучения, отчасти благодаря наличию множества обучающих наборов. В частности, в Национальном институте стандартов и технологии (National Institute of Standards and Technology — NIST) США была создана база данных из 60 000 рукописных цифр с их расшифровками.

Используемая в данной работе база данных MNIST была построена на основе NIST из двух различных баз данных в NIST. Первый набор - SD-1 был собран среди учащихся средней школы, и эти данные намного чище и их легче распознавать, чем набор данных SD-3, собранных среди сотрудников переписи населения США. База данных MNIST собрана путем смешивания этих двух наборов - таким образом, результаты экспериментов с ней не зависят от конкретной обучающей выборки, и мы можем получить разумные результаты.

Данные представлены в виде изображений размером 28x28 пикселей в градации серого. Каждый бит изображения - число от 0 до 255.

Постановка задачи

Для решения задачи классификации рукописных цифр был выбран алгоритм KNN - метод классификации, основанный на поиске k ближайших соседей (K-Nearest-Neighbors). Выбор сделан в пользу именно этого алгоритма, поскольку он, несмотря на свою простоту, дает приемлемые результаты, а так же не требует дополнительного времени на обучение. Так же этот метод позволяет удобно варьировать понятие "близости", то есть дает возможность безболезненно выбирать различные метрики, что позволяет провести большое количество экспериментов с ним.

Предлагаемый метод решения

Программа написана на языке Python, позволяющем легко реализовывать различные алгоритмы машинного обучения.

Роль "точки", метку (цифру от 0 до 9) которой предстоит определить алгоритму, выполняет класс Point, содержащий все пиксели в поле-матрице cs и правильную метку, проставленную создателями базы данных NIST в поле label.

Для тестирования метода KNN на базе данных MNIST в качестве метрики было выбрано расстояние Хемминга, определенное следующим образом:

```
def metr(self, p, pres):
    dist = 0
    for i in xrange(len(self.cs)):
        for j in xrange(len(self.cs)):
            if abs(p.cs[i][j] - self.cs[i][j]) > pres:
                dist += 1
    return dist
```

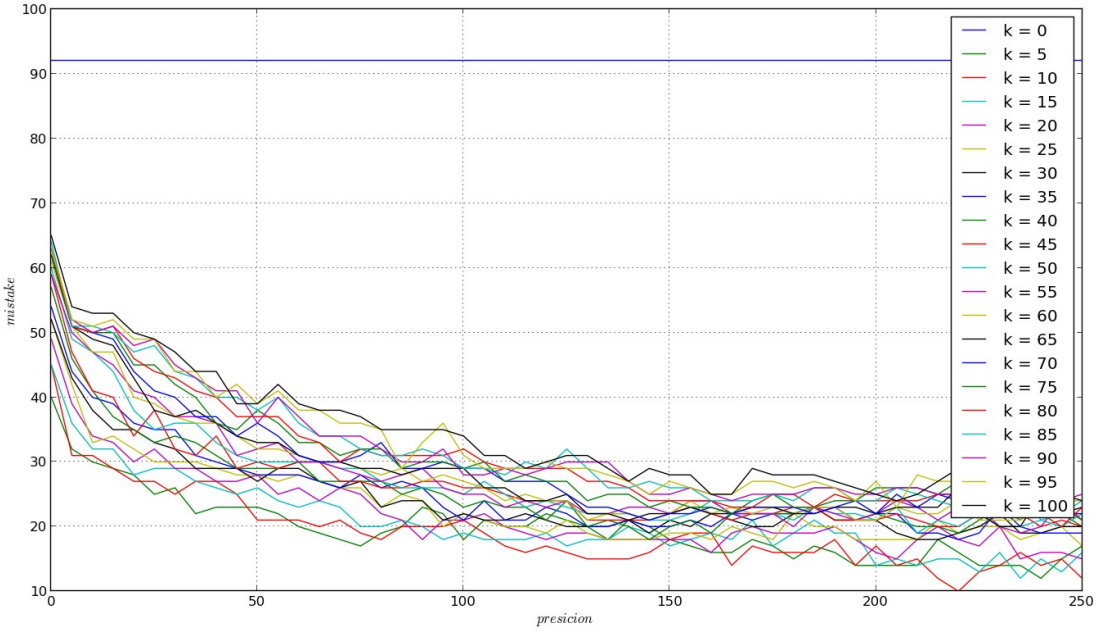
pres в данном случае - точность определения "одинаковости" для двух пикселей. Этот параметр варьировался в ходе экспериментов, для определения наиболее приемлемой метрики (дающей как можно меньшую ошибку).

Так же в ходе экспериментов менялось значение *k* - количество точек-кандидатов, среди которых общим голосованием выбиралось значение метки для определяемой точки.

Результаты

Было проведено четыре эксперимента, которые позволили последовательно установить лучшие параметры алгоритма для классификации рукописных цифр.

1) Первый эксперимент дал грубую оценку для оптимальных значений *k* и *pres*. Программа была запущена на различных значениях *k* (от 0 до 100 включительно с шагом 5) и различных *pres* для каждого *k*. Каждый раз обрабатывалось всего 100 точек в тестовой выборке, а обучающая выборка состояла из 1000 точек, так как большая точность на данном этапе не требовалась. Результаты представлены на графике:

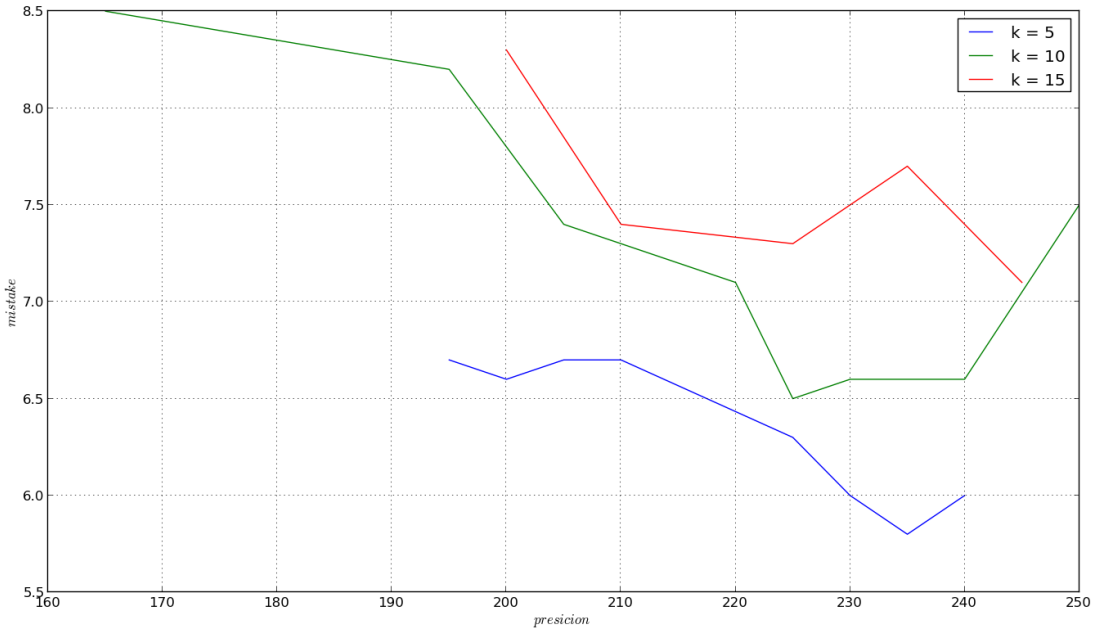


Можно заметить, что при *k* = 0 количество ошибочно определенных цифр не

меняется при изменении pres и составляет порядка 90%. Это ожидаемый эффект, призванный лишь подтвердить равномерность распределения данных, ведь в таких условиях алгоритм дает фиксированный результат (в данном случае ставит метку 0).

Так же по графику видно, что лучший результат дали следующие параметры: k = 10, *pres* = 220. Ошибка в этом случае составила 10%. Однако имелись и близкие к этому результаты параметров, дающие ошибки в 12, 13 и 14 процентов.

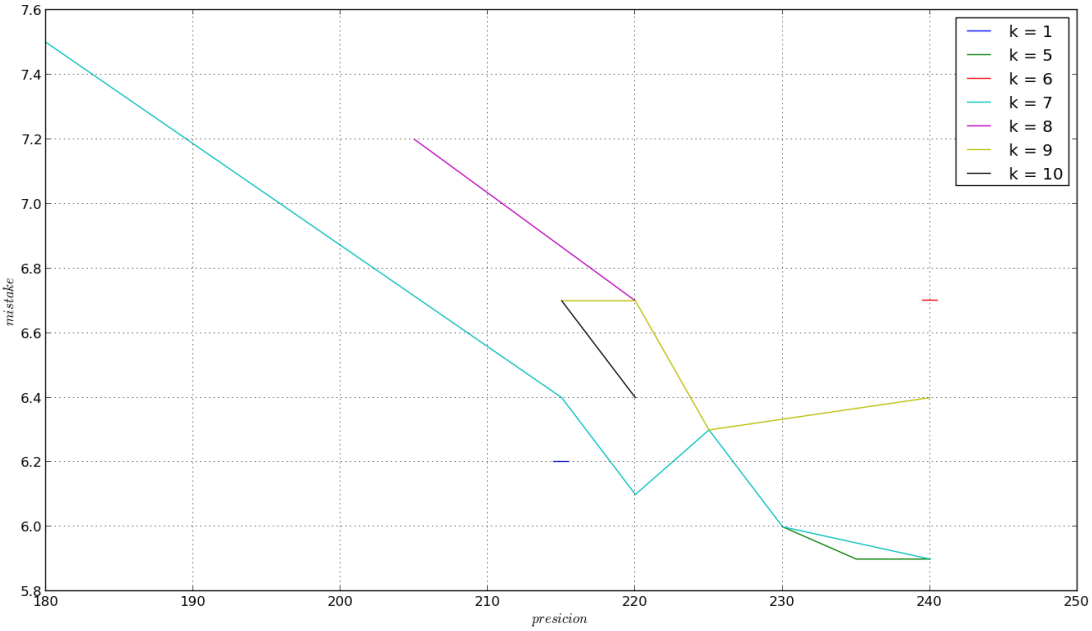
2) Для k и *pres*, соответствующих лучшим значениям предыдущего эксперимента, был проведен более тщательный эксперимент - теперь проверялось по 1000 точек на обучающей выборке из 10000 значений. Результат:



Как видно, чем меньше k, тем классификатор в этом эксперименте был точнее. В частности, лучший результат дали значения k = 5 и *pres* = 235 - ошибка здесь 5.8%. Этот график дал основания искать лучшие k и *pres* среди маленьких значений k и больших значений *pres*, что привело к третьему этапу эксперимента.

3) Ниже показан результат тестирования программы на 100 точках, с обучающей выборкой в 1000 точек, однако, в отличие от первого этапа, k были взяты от 0 до 10 включительно с шагом один, значения *pres* же взяты с тем же шагом 5, но начиная от 150. Результаты представлены на графике ниже. Очевидно, лучший результат показал алгоритм KNN при k = 7 и *pres* = 225, здесь 9% ошибок. Однако, результатов, близких к этому, достаточно, чтобы произвести те же вычисления еще раз, но на большей выборке.

4) Итак, для нескольких самых лучших результатов этапа 3 был проведен эксперимент с большей выборкой точек (10000 точек в обучающей и 1000 в тестовой выборке). Результат:



Для получения итогового результата алгоритм был запущен при трех лучших значениях предыдущих экспериментов на обучающей выборке в 30000 точек и тестовой выборке в 5000 точек. Результаты приведены в таблице:

№п/п	k	pres	mist
1	5	235	4.76
2	5	240	4.92
3	7	240	5.02

Выводы

Результаты экспериментов с различными параметрами для алгоритма KNN в условиях решений задачи распознавания рукописных цифр показывают, что

1) Достаточно брать k = 5 для получения хорошего результата. Это значит, что можно уменьшить время выбора точек-кандидатов до O(n) против O(nlogn) для обычной сортировки сравнениями.

2) Подсчитывая расстояние Хэмминга между точками необходимо считать разными пиксели, на самом деле далеко отстоящие друг от друга (в нашем случае – пиксели контрастных цветов). В данной работе, когда максимальное расстоянием между пикселями было равно 255, идеальным оказалось расстояние 235.

3) Алгоритм KNN при оптимальных значениях k и оптимальной же функции, выбранной в качестве метрики, дает ошибку порядка 4.8% для классификации рукописных цифр базы данных MNIST.

Список литературы

1. http://www.rriai.org.ru/prakticheskiy-primer-raspoznavanie-rukopisnyih-tsifr-5.html - описание нескольких алгоритмов решения задачи классификации рукописных цифр.

2. http://jeremykun.com/2012/08/26/k-nearest-neighbors-and-handwritten-digit-classification/ - пример использования и более подробное описание алгоритма KNN

3. http://yann.lecun.com/exdb/mnist/ - база данных MNIST