



УНИВЕРСИТЕТ ИТМО

Курсовой проект по машинному обучению

Milestone

Максим Слюсаренко, Виктор Шатров,
Григорий Шовкопляс

Постановка задачи

- ▶ Выбрать и реализовать алгоритм для распознавания рукописных цифр
- ▶ Сравнить результаты с другими существующими алгоритмами

Dataset

- ▶ Dataset MNIST
- ▶ Самый известный Dataset с рукописными цифрами
- ▶ Training set на 60000 элементов
- ▶ Test set на 10000 элементов

Известные алгоритмы решения задачи. Линейный классификатор

1. Самый простой метод
2. При этом естественно, что самый неточный
3. Одноуровневый перцептрон получает 8.4% ошибок
4. Попарный линейный классификатор получает 7.6% ошибок

Известные алгоритмы решения задачи. Метод KNN

1. Метод K ближайших соседей
2. Самая простая реализация получает 5% ошибок
3. Самая лучшая реализация с нелинейной деформацией получает 0.52% ошибок

Известные алгоритмы решения задачи. Нейронные сети

1. Нейронные сети от 2 до 6 слоев
2. Получают от 0.35 до 4.7 % ошибок в зависимости от количества слоев, функции потерь и других настроек

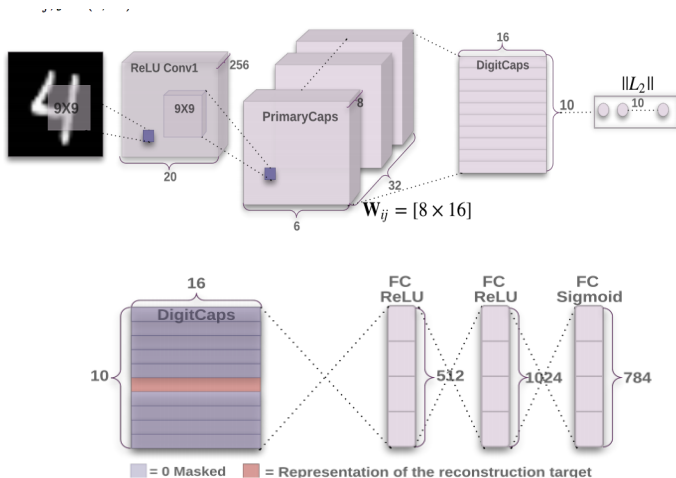
Известные алгоритмы решения задачи. Сверточные нейронные сети

1. Сверточные нейронные сети с различной архитектурой и различным количеством слоев
2. Получают от 0.23 до 1.7 % ошибок
3. Некоторые алгоритмы требуют нормализации по ширине

Что хотим использовать для решения задачи?

- ▶ Капсульные нейронные сети
- ▶ Используют немного другую архитектуру по сравнению с обычными или сверточными нейронными сетями
- ▶ Более подробное описание в следующих слайдах

Капсульные нейронные сети. Архитектура



Капсульные нейронные сети. Глобальное описание

- ▶ Сначала просто происходит свертка, чтобы получить из картинки матрицу признаков
- ▶ Потом по матрице признаков делается много сверток и получаем слой из 32 сверток
- ▶ Каждая капсула возвращает не единственное число, как нейрон, а некоторый вектор
- ▶ При соединении двух капсульных слоев происходит процедура Dynamic Routing
- ▶ По последнему слою происходит обучение, чтобы длина правильного вектора была как можно больше, а длины неправильных векторов как можно меньше

Капсульные нейронные сети. Dynamic Routing

- ▶ Идея состоит в том, чтобы за несколько итераций выделить капсулы предыдущего уровня, которые сильнее всего влияют на результат, то есть попадают ближе к центру текущей капсулы, после чего дать им больше веса в будущем
- ▶ Алгоритм:

Procedure 1 Routing algorithm.

```
1: procedure ROUTING( $\hat{\mathbf{u}}_{j|i}, r, l$ )  
2:   for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow 0$ .  
3:   for  $r$  iterations do  
4:     for all capsule  $i$  in layer  $l$ :  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$  ▷ softmax computes Eq. 3  
5:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$   
6:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$  ▷ squash computes Eq. 1  
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$   
   return  $\mathbf{v}_j$ 
```

Капсульные нейронные сети. Формулы в алгоритме Dynamic Routing

- ▶ 4 строка:

$$c_{ij} = \frac{\exp(b_{ij})}{\sum \exp(b_{ik})}$$

- ▶ 6 строка:

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|^2}$$

- ▶ Значение $\hat{u}_{j|i}$ вычисляется перемножением выхода предыдущего шага на матрицу весов W_{ij} следующим образом:

$$\hat{u}_{j|i} = W_{ij} u_i$$

Капсульные нейронные сети. Loss Function

- ▶ Оптимизируется следующая функция:

$$L_c = T_c \max(0, m^+ - \|v_c\|)^2 + \lambda(1 - T_c) \max(0, \|v_c\| - m^-)^2$$

- ▶ В этой функции $T_c = 1$ тогда и только тогда, когда цифра класса c присутствует
- ▶ $m^+ = 0.9$
- ▶ $m^- = 0.1$
- ▶ $\lambda = 0.5$

Капсульные нейронные сети. Заключение

- ▶ Утверждается, что данный алгоритм превосходит сверточные нейронные сети за счет того, что не происходит стадии `max pool`, на которой убирается информация, чтобы уменьшить размерность
- ▶ Также утверждается, что при некотором повороте входных данных, данный алгоритм работает проще и интуитивно понятнее, чем сверточные нейронные сети

Наши результаты

- ▶ Процент ошибок на MNIST = 0.72 %
- ▶ Не такой большой процент, так как обучали сравнительно небольшое время
- ▶ Код на github = <https://github.com/MaximSlyusarenko/capsNet>



УНИВЕРСИТЕТ ИТМО

Спасибо за внимание!