Министерство образования и науки Российской Федерации
Санкт-Петербургский государственный технический университет
Институт прикладной математики и механики
Кафедра «Телематика»

ЛАБОРАТОРНАЯ РАБОТА

ПО ТЕМЕ

«Машинное обучение»

по направлению 02.04.01.02 «Организация и управление суперкомпьютерными системами»

Выполнил:
Студент гр. 13643.1 Титов А.И.
Проверил: Уткин Л.В.

Санкт-Петербург
2019

Оглавление

1 Лекция №1. 12.02.2019 **3**
1.1 Сверточные сети. 3
1.2 Max-pooling 3

2 Лекция 2 26.02.19 **4**
2.1 7 приемов для более эффективного обучения НС 4
2.1.1 Прием №1 Stochastic Versus Batch Learning 5
2.1.2 Прием №2 Shuffling the Examples (Перемешивание) 5
2.1.3 Прием №3 Normalizing the Inputs (нормализация) 5
2.1.4 Прием №4 Sigmoid 5
2.1.5 Прием №5 Choosing Target Values 5
2.1.6 Пример №6 Initializing the Weights 6
2.1.7 Пример №7 Choosing Learning Rates 6

3 Лекция 3 5.03.19 **6**
3.1 Сегментация изображений 6

Постановка задачи **6**

1 Лекция №1. 12.02.2019

1.1 Сверточные сети.

Вход - большой размерности: каждый нейрон имеет огромное число соединений. Малая картинка 100x100 пикселей (размерность входа 10000), каждый нейрон имеет 10000 параметров. Если скрытый слой - 2000 нейронов, то всего 2 x 10^7 соединений.

$$f(w, x) \to y \in \{1, 2, 3, 4\}$$

x - вектор характеризующий параметры.

w - вектор весов (соединений).

Скрытый слой содержит больше нейронов, чем изначальный.

Чтобы научить каждый вес требуется 3 - 5 изображений (при том картинки размеченные, то есть уже разбиты на классы, например “кошки”, “собаки”).

То есть требуется действительно большое количество картинок для обучения большой сетки.

а что, если часть соединений убрать?

ЛеКун в 1995 предложил устроить все по аналогии работы глаза. Обрабатывать не все изображение, а отдельные “квадраты”.

Как уменьшить число соединений? - Сделать часть весов одинаковыми (“weight sharing” или **свертка**) - w1 = w4 = w7, w2 = w5 = w8, w3 = w6 = w9 - вместо хранения всех весов, храним w1 w2 w3

Вход $\Rightarrow$ Свертка $\Rightarrow$ Пулинг (subsampling) $\Rightarrow$ Свертка $\Rightarrow$ Пулинг $\Rightarrow$ … $\Rightarrow$ Свертка $\Rightarrow$ Пулинг $\Rightarrow$ Выход

Сверточный слой - реализует обычную операцию свертки, двигаясь по изображению скользящим окном.

Пулинговый слой - сжатие данных для достижения меньшей размерности.

$$\sum w_{ij} a_{ij}$$

$_{ij}$ - элементы ядра входного слоя (значение квадратов пикселей на входном изображении). A - матрица.

w_{ij} - элементы ядра сверточного слоя. W - матрица.

1.2 Max-pooling

получено изображение от сверточного слоя. Разбиваем опять сеткой это изображение и строим новое, основываясь на максимальном значении в окне сетки.

Были значения на иображении:

```
0.0   0.0   0.0   0.0
0.0   0.0   0.0   0.5
0.0   0.0   0.5   1.0
1.0   0.5   0.0   0.0
```

Разбиваем:

```
0.0   0.0   |  0.0   0.0
0.0   0.0   |  0.0   0.5
------------------------
0.0   0.0   |  0.5   1.0
1.0   0.5   |  0.0   0.0
```

Производим Max-pooling:

```
0   0.5
1   1
```

AVE (Average-pooling) - тоже самое только рассматриваются средние значения.

2 Лекция 2 26.02.19

Преимущества сверточных сетей:

- один из лучших для распознания и классификации
- меньшее количество весов по сравнению с нейронной сетью
- обобщает информацию, а не попикселльно запоминает каждую картинку в коэффициентах. (спорное утверждение)
- распараллеливание, возможно реализация на ГПУ.
- устойчивость к деформации изображения (поворот, сдвиг)
- обучение при помощи классического метода обратного распространения ошибки

Недостатки сверточных сетей:

- По большей части архитектура для распознавания изображений
- Слишком много варьируемых параметров. А обучение итак слишком долгое, поэтому обучение слишком затруднено, когда существует большое множество вариантов параметров.

2.1 7 приемов для более эффективного обучения НС

Напомним:

- Требуется определить параметры НС, такие как выборки, скорость и т.д.;
- Нет четких правил дл выбора
- 1. Обучение - минимизировать функцию потерь.
  2. Обобщение - прогнозирование на новых примерах.

  Требуется найти компромисс между обучением и обобщением.
- Для хорошего обучения требуется оценивать качество обучения, мы можем это сделать с помощью:

  1. Смещение - требуется сделать так, чтобы данные имели разброс ближе либо к 0 либо к 1
  2. Дисперсия - мера того, насколько выход НС варьируется для разных данных. Дисперсия должна быть как можно более меньшей.

Чем дольше мы обучаем тем большая дисперсия, но малое смещение. И наоборот, соответственно. Это происходит потому что НС начинает учиться на шуме.

2.1.1 Прием №1 Stochastic Versus Batch Learning

Градиентный спуск и обратное распространение ошибки заключается в том, что если есть сетка (не важно сверточная или нет) с весами $w_1 \dots w_n$, а на вход подаем вектор $x_1 \dots x_n$ и вычисляется $S = y^* - y$ для каждого x. (что-то пропустил, думаю надо будет погуглить)

- Стохастический градиентный спуск - из обучающей выборки выбирается один элемент на каждой итерации.
- Пакетный - просматривается и модифицируется вся сетка.

Stochastic Learning и Mini-Batch

- Обычно намного быстрее, чем пакетное обучение;
- Часто приводит к лучшим решениям;
- Может быть использовано для отслеживания изменений

Но шум все может испортить
=> Mini-Batch что-то там и если этот параметр больше то хуже.

2.1.2 Прием №2 Shuffling the Examples (Перемешивание)

НС обучается быстрее на наиболее неожиданных примерах.

С точки зрения градиента - неожиданный пример портит картину. И это работает как мутация. То есть можно ускорить обучение с помощью перемешивания выборов из разных классов. Возможен вариант алгоритм, когда выбор каждого объекта неравновероятен, причем вероятность выпадения объекта обратно пропорциональна величине ошибки на объекте.

2.1.3 Прием №3 Normalizing the Inputs (нормализация)

Было замечено, что среднее значение каждой входной из обучающей выборки ближе нулю, то обучение проходит лучше. То есть требуется для обучающей выборки сделать мат.ожидание 0. То есть лучше вычислить сначала среднее, а потом вычесть из каждого входного данного полученное значение.

Масштабирование ускоряет обучение. Вычислим среднеквадратичное отклонение $\sigma = \frac{1}{n}\sum(x_i - x_{ave})^2$.

$x_i^* \leftarrow \frac{x_i - x_{ave}}{\sqrt{\sigma^2 + \epsilon}}$

также производится декореляция.

2.1.4 Прием №4 Sigmoid

(существуют сигмоиды, бисигмоиды и релу (проблема релу - чем более функция активации линейна, тем сложнее достигнуть нелинейности).)

Симметричные сигмоиды часто сходятся быстрее, чем стандартная логистическая функция.

2.1.5 Прием №5 Choosing Target Values

- Что выбрать {-1,1} или {0,1}?
- Нужно брать целевые значения - в точке максимальной второй производной на сигмоиде (например {0.1,0.9} вместо {0,1}).

2.1.6 Пример №6 Initializing the Weights

Веса должны выбираться так, чтобы сигмоид активировался в линейной области. Тоже самое и для ReLu.

2.1.7 Пример №7 Choosing Learning Rates

- Уменьшают скорость обучения когда весовой вектор “колеблется”, и увеличивают, когда он устойчив.
- Различная скорость обучения для каждого веса может улучшить сходимость.
- скорость обучения должна быть пропорциональна квадратному корню из числа входов в нейрон.
- Веса в нижних слоях обычно должны быть больше, чем в более высоких слоях.

3 Лекция 3 5.03.19

3.1 Сегментация изображений

Хорошим подходом является добавление дополнительного шага в алгоритм распознания изображений перед классификацией объекта. Этот шаг - сегментация. По-сути идея такая - выделить сначала сегменты изображения которые нас интересуют, а потом уже классифицировать объекты в сегментах.

Постановка задачи

Рис. 1: Что такое вайфу?

See table 1.

Таблица 1: Вроде как таблица

Col1	Col2	Col3
Left-aligned	Centered	Right-aligned

Col1	Col2	Col3
blah	blah	blah

Какой-то текст

SININ fndjksk

```
#include<iostream>
using namespace std;

int main() {
    cout << "I shoud do my exam tasks" << endl;

    return 0;
}
```