

Введение в нейронные сети

Арсений Горин

МГТУ, 21 ноября 2016

Содержание

1 Виды сетей и общие понятия

2 Полносвязная нейронная сеть

- Общие моменты
- Обучение сетей
- Немного практики

3 Сверточные сети

- Общая идея
- Свертка и фильтры
- Нелинейность и пулинг
- Примеры и немного практики

Глубинное обучение (Deep Learning) - методы обучения нейронных сетей с большим количеством параметров (**Deep Neural Networks: DNN**).

- Доминирующая модель распознавания изображений, речи, машинного перевода, интеллекта, ...
- Особенно эффективны при работе с большими данными (large capacity models)
- Сегодня можно строить очень сложные (и весьма точные) модели в домашних условиях (видео карты с графическими процессорами GPU)
- Обучение требует некоторых экспертных знаний, основу которых постараемся сегодня рассмотреть

Содержание

1 Виды сетей и общие понятия

2 Полносвязная нейронная сеть

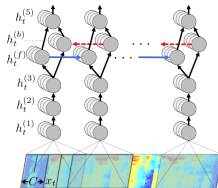
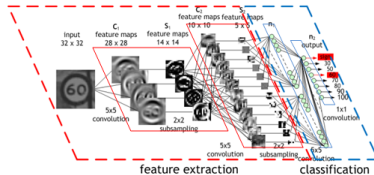
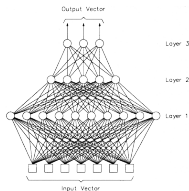
- Общие моменты
- Обучение сетей
- Немного практики

3 Сверточные сети

- Общая идея
- Свертка и фильтры
- Нелинейность и пулинг
- Примеры и немного практики

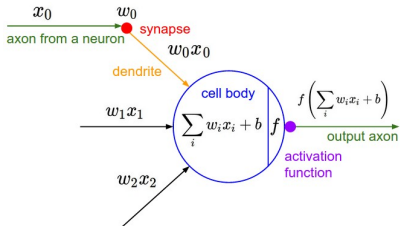
Основные виды сетей

- **Полносвязные (fully connected) или многослойный персептрон (MLP)**
 - Комбинация простых линейных моделей и нелинейных преобразований
 - Не использует дополнительных знаний о данных (все в кучу)
 - Самая ранняя архитектура DNN
- **Сверточные (convolutional)**
 - Умеет извлекать интерпретируемые признаки
 - Доминирующая модель компьютерного зрения
- **Рекуррентные (recurrent)**
 - Классификатор последовательности (умеет моделировать время)
 - Отлично работает в распознавании речи, машинном переводе

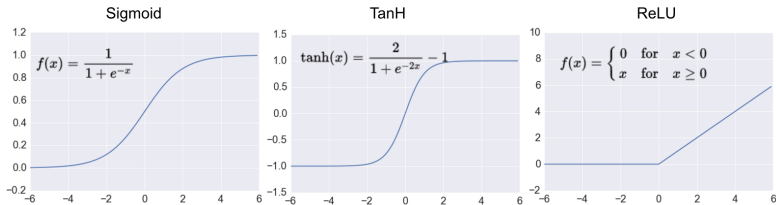


Упрощенная модель нейрона

- $X = \{x_i\}$ - признаки объекта, или выходы с других нейронов
- веса $W = \{\omega_i\}$ - линейная модель
- f - нелинейное преобразование (ниже подробнее)
- Выход - активация (часто 0/1)



Функция активации - как нейрон реагирует на входной импульс



Обучение нейрона - настройка весов W по данным (объект-признак / ответы)

Содержание

1 Виды сетей и общие понятия

2 Полносвязная нейронная сеть

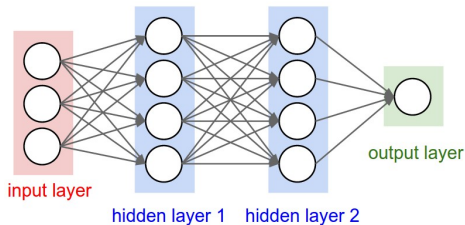
- Общие моменты
- Обучение сетей
- Немного практики

3 Сверточные сети

- Общая идея
- Свертка и фильтры
- Нелинейность и пулинг
- Примеры и немного практики

От нейрона к нейронной сети

- Входной слой - вектор признаков (яркость пикселей, размеры лепестков Ириса, и т.п.)
- Скрытые слои - соединенные между собой нейроны (в реальных сетях $10^6 - 10^7$ нейронов, у человека $\sim 10^{10}$)
- Выходной слой - ответ (кошка/собака, тип цветка, и т.п.)



Сеть называется **полносвязной**, т.к. каждый нейрон связан со всеми нейронами предыдущего слоя. Такая сеть еще называется **сетью прямого распространения (feed forward network)**

Обучение. Градиентный спуск

Обучить DNN значит **подобрать** такие **веса** для каждого нейрона, при которых сеть выдает **минимальную ошибку**

Используют **алгоритм градиентного спуска (gradient descent)**

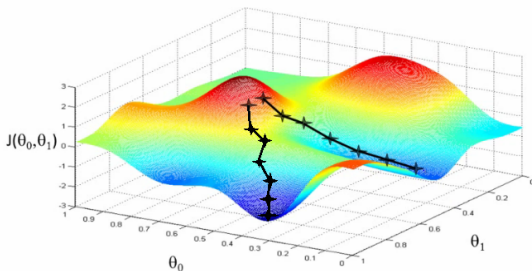
- 1 Задаем случайные веса с небольшим разбросом
- 2 “Прогоняем” через сеть объекты обучающей выборки
- 3 Считаем ошибку ответов сети E (например, среднюю квадратическую)
- 4 Используя метод обратного распространения ошибки (back propagation), считаем ошибку каждого отдельного веса $\frac{dE}{dw_i}$
- 5 Обновляем значения весов $w_i = w_i - \alpha \frac{dE}{dw_i}$
- 6 Повторяем шаги 2-5 пока уменьшается ошибка на тестовых данных

Параметр α называется **learning rate**, подбирается эмпирически

Обучение. Градиентный спуск

Градиентный спуск:

- позволяет найти **локальный минимум** (на практике работает)
- важно **контролировать ошибку** на независимых данных
- на практике используем **стохастический градиентный спуск (stochastic gradient descent)**: каждое обновление весов делается по небольшой подгруппе объектов (**minibatch size**: 10...500)



Основные задачи DeepLearner'a

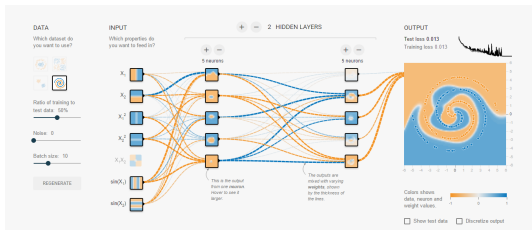
- В классическом машинном обучении большая часть отдавалась выбору признаков (**feature engineering**)
- Нейронные сети сами выделяют признаки из достаточно сырых данных

От **feature engineering** к **architecture engineering**

- Как выбрать архитектуру (тип сети, нелинейность, число параметров)
- Оптимизация: выбор learning rate, алгоритмы обучения, устойчивость
- Быстродействие моделей и память

Как это выглядит на практике

Интерактивный интерфейс библиотеки tensorflow от google
<http://playground.tensorflow.org>



Задачи:

- Запустить пример по умолчанию и посмотреть что происходит
- Эксперимент с персептроном Розенблатта, или то, почему про нейронные сети забыли на 20 лет (XOR классификация)
- Проблема переобучения, или почему DeepLearning требует настройки

Содержание

1 Виды сетей и общие понятия

2 Полносвязная нейронная сеть

- Общие моменты
- Обучение сетей
- Немного практики

3 Сверточные сети

- Общая идея
- Свертка и фильтры
- Нелинейность и пулинг
- Примеры и немного практики

Сверточные нейронные сети

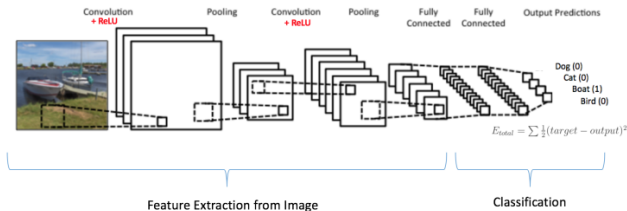
Сверточные нейронные сети (convolutional neural networks) - на сегодняшний день самые передовые модели компьютерного зрения и других областей ИИ.

Основные “исторические моменты”:

- 1990 (Yann LeCun) LeNet: первое успешное применение сверточной сети для распознавания цифр
- 2012 (Alex Krizhevsky) AlexNet: победа CNN в соревнованиях ImageNet с огромным отрывом, рост интереса к моделям CNN и DeepLearning в целом
- Современные сети имеют зачастую меньше параметров, чем AlexNet, работая гораздо точнее

Общая архитектура CNN

- Обработка от общего к частному (глубже слой - детальнее признаки)
- Основной блок - **фильтры свертки (convolutional filter)**
- Результат проходит через **нелинейное преобразование (ReLU)**
- Уменьшение размерности - **пулинг (max pooling)**
- **Классификатор** - **полносвязная НС**, использующая признаки, найденные верхними слоями



Операция свертки (convolution)

- Картинку представляем в виде **матрицы яркости** (3 матрицы, если картинка цветная)
- **Фильтр** - матрица небольшого размера (обычно 3x3, 5x5, 8x8)
- **Признак** - **свертка двух матриц** (фильтр скользит по картинке, считаем скалярное произведение окна и матрицы фильтра)

8	2	0	2	0	6	2	6	2	0	2	0	2	0	2
6	2	0	2	1	12	2	1	3	137	37	0	152	147	34
6	2	1	3	0	4	138	253	235	256	255	229	256	11	15
6	2	0	16	9	5	152	251	45	21	154	159	154	251	233
16	2	0	2	0	0	0	145	146	3	10	2	11	224	253
6	2	2	2	4	15	235	216	0	2	34	168	247	244	156
1	2	2	2	0	0	253	255	23	32	254	241	255	161	2
6	2	0	4	0	1	252	256	228	235	255	234	112	25	2
6	2	1	4	0	21	251	253	251	255	172	31	6	2	1
3	2	4	3	163	225	221	255	233	130	8	0	2	3	11
2	2	21	162	255	254	255	125	6	2	10	14	5	2	3
3	23	242	255	141	68	255	245	123	7	2	0	2	5	2
26	251	237	58	0	57	251	255	144	0	0	0	2	7	2
125	255	141	2	87	24	255	256	3	0	2	13	1	2	1
155	248	235	116	235	255	141	34	3	11	2	1	2	3	2
85	257	253	241	255	216	21	1	2	1	2	0	2	4	3
6	23	112	257	114	58	2	2	3	0	2	0	2	3	7
6	2	2	2	0	6	2	2	3	0	2	0	2	2	3

1	0	1
0	1	0
1	0	1

Упрощенный пример свертки

Путь изображение и фильтр кодируются 1 и 0 (черный-белый).

Изображение

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Фильтр

1	0	1
0	1	0
1	0	1

Результат свертки - новая матрица (**feature map**):

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4		

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	4

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	4
2		

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	4
2	4	

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	4
2	4	3

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0





4	3	4
2	4	3
2		

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	4
2	4	3
2	3	

Примеры фильтров

Несколько стандартных фильтров из пакета photoshop

Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

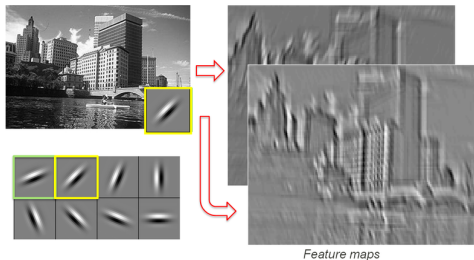
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Black magic

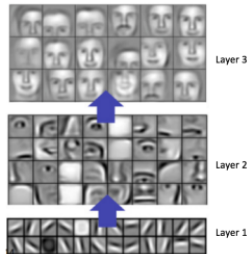
С помощью градиентного спуска **можно выучить оптимальные параметры фильтров**.

То есть система, наблюдая данные, сама учится, какие признаки являются наиболее полезными для решения конкретной задачи классификации.

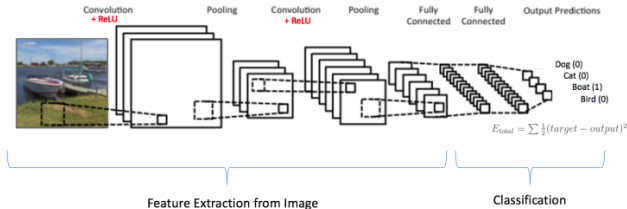
Пример применения фильтра



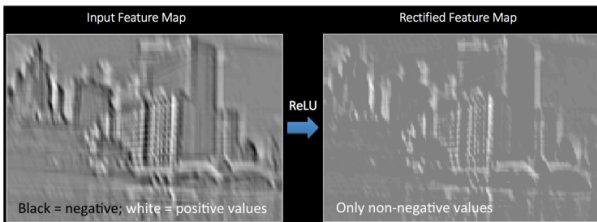
Feature maps в многослойной сети



ReLU и зачем он нужен

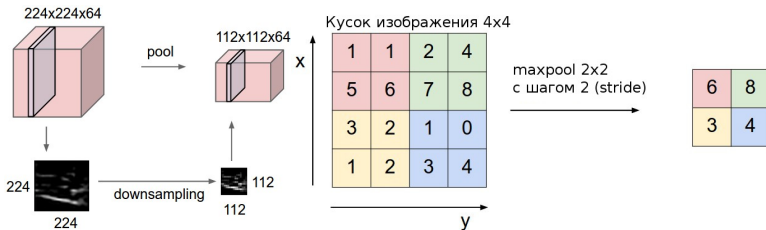


Изображение, прошедшее через банк фильтров, попиксельно проходит через **нелинейное преобразование (ReLU - rectified linear unit)**.
Наличие ReLU позволяет сети выучивать нелинейные зависимости признаков.
ReLU по сути выбрасывает все отрицательные значения



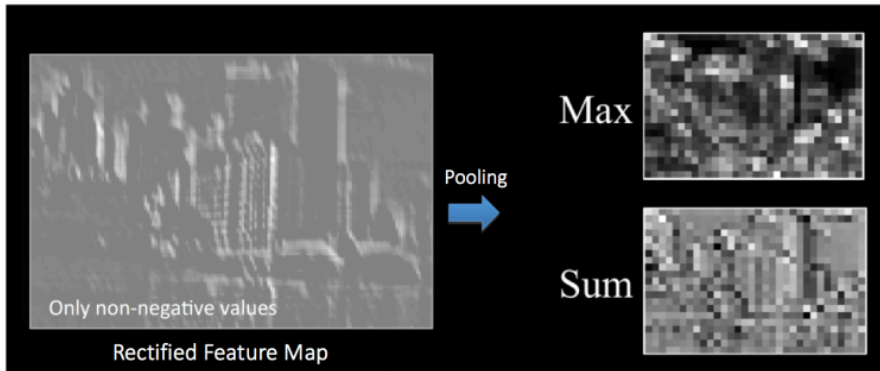
Pooling (или subsampling)

- В реальных сетях первый слой состоит из **порядка 100 фильтров**, каждый из которых создает обработанную копию картинки (feature map)
- Задача пулинга** - уменьшение размерности
 - Меньше вычислений
 - Меньше использования памяти
 - Устойчивость к шумам

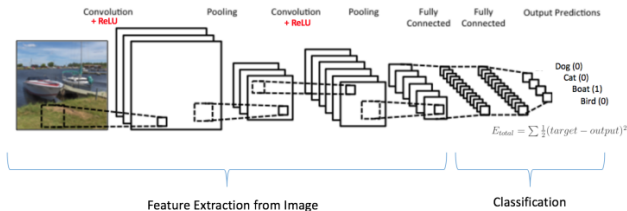


Результат пулинга

ReLU и MaxPool - фиксированные операции (т.е., учить их не нужно)



Последняя часть - классификация



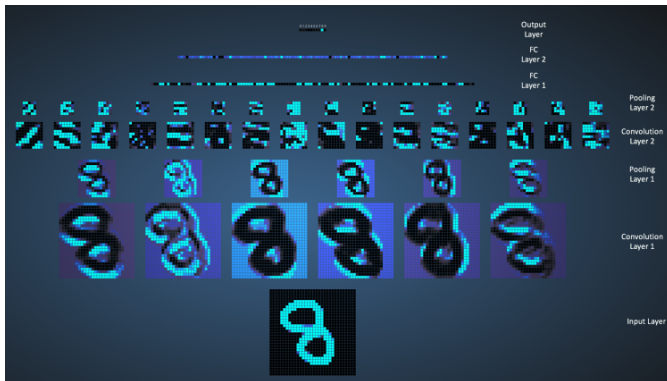
- После обработки фильтрами, ReLU и пулингом, параметры записываются в вектор действительных чисел - признаки.
- Признаки подаются на вход классификатору - полносвязной нейронной сети



На практике

Визуализация сети, классифицирующей MNIST цифры

<http://scs.ryerson.ca/~aharley/vis/conv/flat.html>



Еще пример (CIFAR-10 классификация)

<http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>