НЕЙРОННЫЕ СЕТИ: ВЫЧИСЛИТЕЛЬНЫЕ АСПЕКТЫ

П.В. Яковлев

p.yakovlev@spbu.ru

24 ноября 2022 г.

Практика машинного обучения (искусственного интеллекта, распознавания образов) прошла огромный путь начиная с середины прошлого века.
Основной скачок в развитии машинном обучении связан прежде всего с

- увеличением мощности компьютеров,
- увеличением количества цифровых данных (цифрового следа) оставляемых человеком в процессе его жизнедеятельности. Причем скорость увеличения накопленных данных постоянно увеличивается. Если в 1995 году общий объем цифровых данных накопленных человечеством оценивается примерно в $130 * 10^9$ гигабайт, то к 2020 году этот объем составил уже около $140 * 10^{12}$ гигабайт.

Вместе с общим ростом накопленных цифровых данных увеличился и объем, так называемых, размеченных данных, необходимых для обучения компьютерного алгоритма распознавания образов. В основном это касается не структурированный данных, таких как изображения, записи звуковых и прочих аналоговых сигналов, показаний медицинских приборов, и т.д.

В рамках настоящего доклада мы будем рассматривать только машинное обучение с “учителем” (Supervised Learning).
Некоторые области применения нейронных сетей для обучения с “учителем” (Supervised Learning), приведены в следующей таблице 1.

Один из ведущих специалистов по машинному обучению Andrew Ng отмечает, что интенсивное развитие нейронных сетей связано со взрывным ростом доступных для исследователей размеченных данных. Он приводит график развития машинного обучения, в том числе “глубоких” нейронных сетей в зависимости от доступности размеченных данных, Рис.1.

Эффективность нейронных сетей начинает проявляться только при наличии достаточного количества размеченных данных. В этом смысле традиционные алгоритмы машинного обучения, такие как СВМ метод, логистическая регрессия и др. актуальны и сегодня и более эффективны, чем нейронные сети, в случае малых обучающих выборок.

---

*Семинар по оптимизации, машинному обучению и искусственному интеллекту «O&ML» http://oml.cmlaboratory.com/

Информация на входе	Информация на выходе	Область применения	Применяемая нейронная сеть
Характеристика апартаментов	Цена	Недвижимость	Стандартная сеть
Информация о потенциальном покупателе	Переход по ссылке на сайт или нет	Реклама	Стандартная сеть
Аудио трек	Расшифровка записи	Распознание речи	Рекуррентная сеть (RNN)
Изображение	Объект из списка	Распознание	Сверточная сеть (CNN)
Текст на английском языке	Текст на русском языке	Машинный перевод	Рекуррентная сеть (RNN)
Изображение, информация с радара	Расположение машин в потоке	Автоматическое вождение	Гибридные сети

Таблица 1.

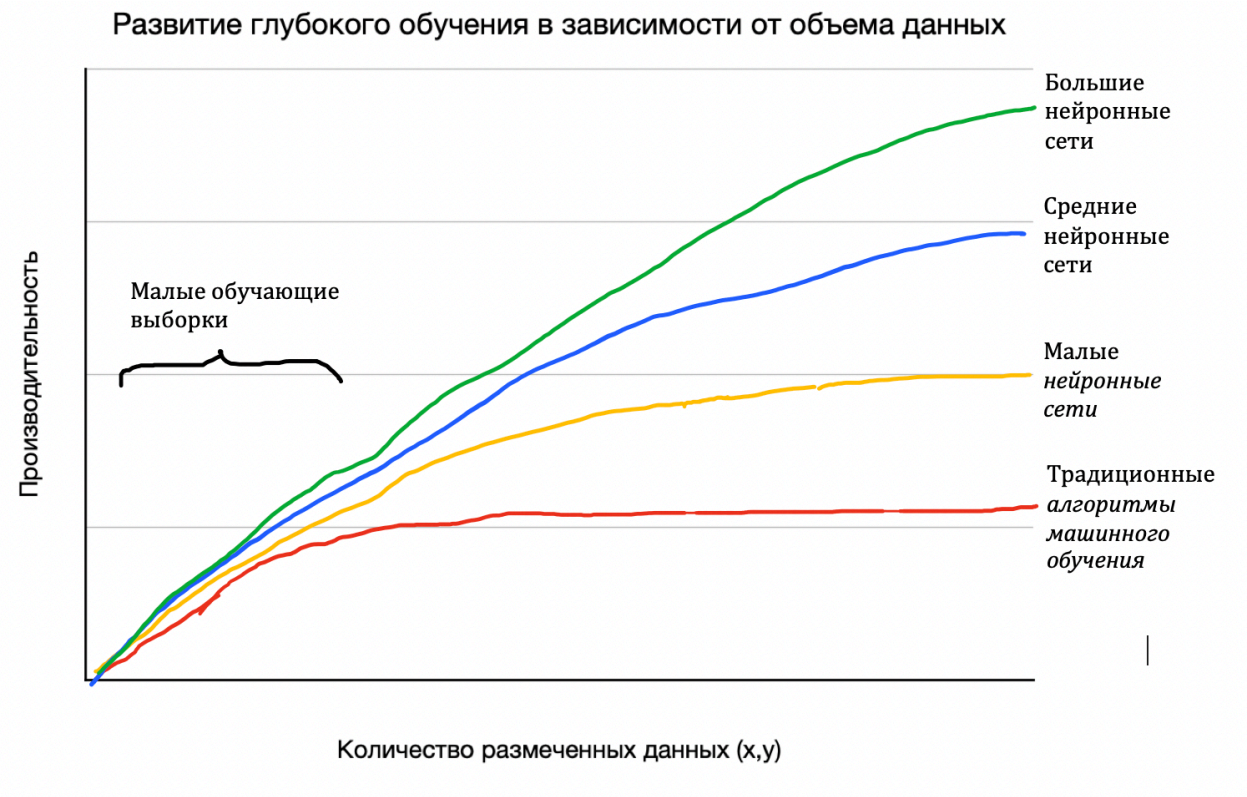


Рис. 1

В настоящем докладе в основном используются материалы обучающего курса по глубокому обучению “Neural Networks and Deep Learning” Стэнфордского университета и “Deep Learning AI”, разработанный Andrew Ng, www.andrewng.org, www.deeplearning.ai.

По мнению автора упомянутого курса Andrew Ng термин “глубокое обучение” является по своей сути неким “брендом”, не имеет строгого определения и связан, в первую очередь, с количеством скрытых слоев в нейронов сети. Выбор структуры нейронной сети является эмпирическим и циклическим процессом. Распространены два эмпирических объяснения почему во многих случаях лучше работают глубокие сети с большим количеством слоем.

1. При прохождении данных от слоя к слою на каждом уровне могут выделяться какие-то свои характерные признаки (черты) исходных данных. Это хорошо прослеживается в сверточных сетях, когда отдельные слои могут выделять свои особенные признаки, Рис. 2.
2. Второй аргумент имеет более формальную основу. Предположим, что нам необходимо вычислить значение исключающего “или” для n двоичных переменных, $y = x_1\ XOR\ x_2\ XOR\ x_3 \ldots XOR\ x_n$. Тогда для того, чтобы вычислить это значение в качестве выхода однослойной нейронной сети потребуется 2^n нейронов. Если же вычислить значение y в виде дерева, Рис. 3, то потребуется всего $O(\log n)$ слоев. Для примера, изображенного на рисунке, это соответственно будет 16 нейронов в первом случае и два слоя и три нейрона во втором.

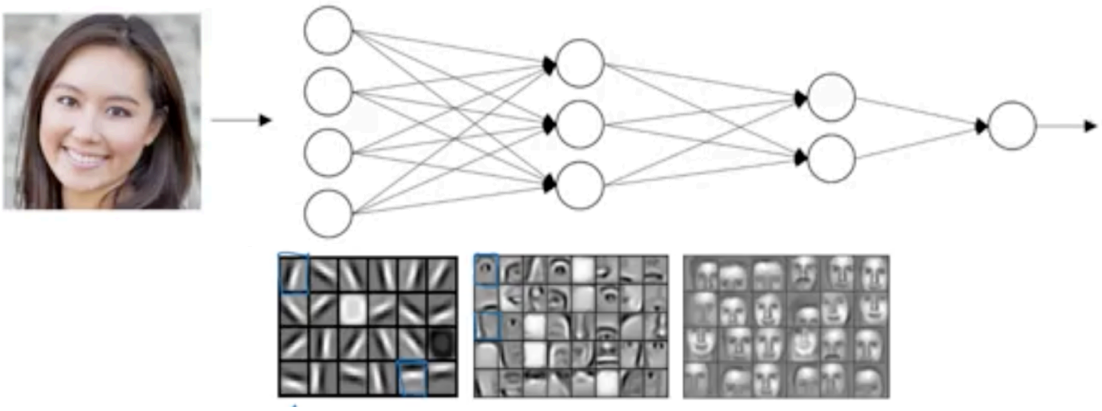


Рис. 2

Рассмотрим три основных вычислительных аспекта реализации машинного обучения с “учителем” при помощи нейронной сети:

1. Эффективная организация вычислений с точки зрения скорости вычислений;
2. Сходимость применяемого метода оптимизации, выбор направления спуска и регулировка шага;
3. Устойчивость вычислений и основные применяемые методы регуляризации.

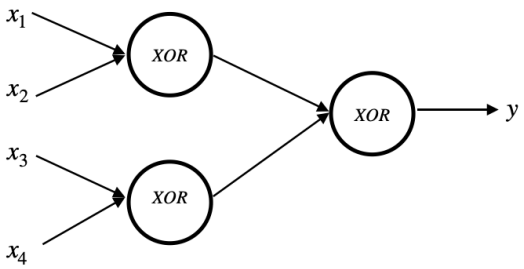


Рис. 3

1. Схема вычислений. Векторизация

Современные программные модули и процессоры позволяют очень эффективно производить матричные вычисления, которые производятся на порядки быстрее, чем прямые вычисления с использованием циклов типа “**for** i = first **to** last **do** statement”.

Структура нейронной сети позволяет избежать таких циклов, за исключением одного цикла при переходе от одного слоя нейронной сети к следующему. Это достигается при помощи так называемой векторизации. Рассмотри, как это организовано на примере простой нейронной сети. На Рис.4а-в изображены структуры простой нейронной сети для нескольких вариантов скрытых слоев. Скрытыми называются слои, которые лежат между входом и выходом нейронной сети.

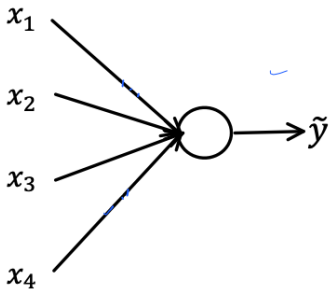


Рис. 4а Логистическая регрессия.

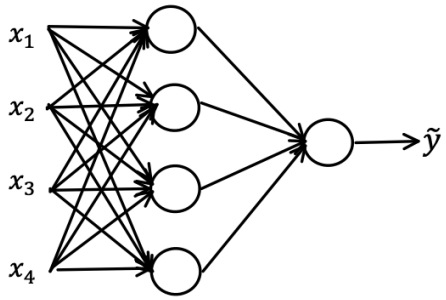


Рис. 4б Один скрытый слой

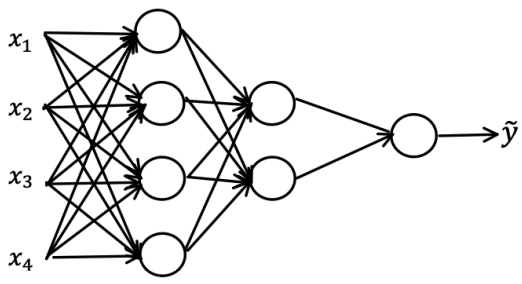


Рис. 4в Два скрытых слоя.

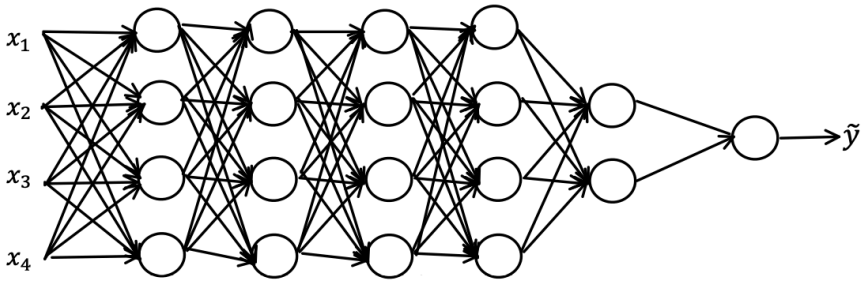
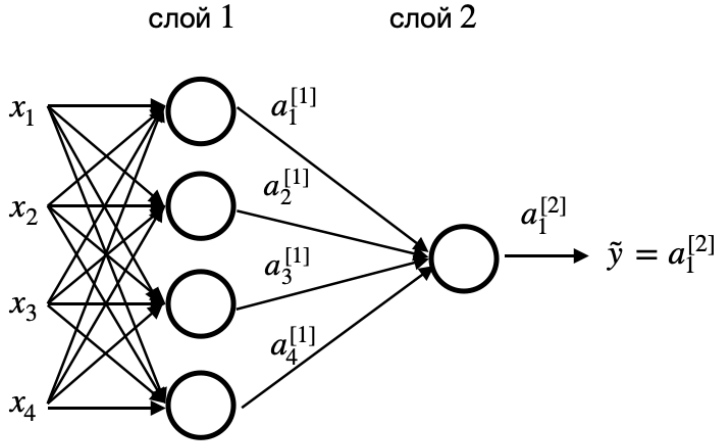


Рис. 4г Пять скрытых слоев

Рассмотрим подробнее как происходят вычисления в каждом слое нейронной сети. На Рис 5. Схематично приведены вычисления в нейронной сети с одним скрытым слоем и четырьмя входами.



$$z_1^{[1]} = (w_1^{[1]})^T x + b_1^{[1]}, \quad a_1^{[1]} = g^{[1]}(z_1^{[1]})$$

$$z_2^{[1]} = (w_2^{[1]})^T x + b_2^{[1]}, \quad a_2^{[1]} = g^{[1]}(z_2^{[1]})$$

$$z_3^{[1]} = (w_3^{[1]})^T x + b_3^{[1]}, \quad a_3^{[1]} = g^{[1]}(z_3^{[1]})$$

$$z_4^{[1]} = (w_4^{[1]})^T x + b_4^{[1]}, \quad a_4^{[1]} = g^{[1]}(z_4^{[1]})$$

$$z_1^{[2]} = (w_1^{[2]})^T x + b_1^{[2]}, \quad a_1^{[2]} = g^{[2]}(z_1^{[2]})$$

Рис. 5

Здесь $w_j^{[l]},\ l \in \{1,2\},\ j \in 1:4$ - весовые коэффициенты нейронов слоя l,
$g^{[l]},\ l = 1,2,$ - функция активации слоя l.

Далее, мы можем записать вычисления для одного слоя и для одного вектора входа из обучающей выборки.

Введем обозначения:

$n = n_x$ - количество входов нейронной сети,

m - количество векторов в обучающей выборке,

$x^{(i)} = x^{(i)}[1:n],\ i \in 1:m$ - вектор входа,

L - количество слоев,

n_l - количество нейронов (входов) в слое $l \in 0:L$,

$a^{[l]} = a^{[l]}[1:n_l]$ - вектор активации слоя $l \in 0:L$, активация нулевого слоя есть вектор входа, $a^{[0]} = x$, активация последнего слоя L - выход сети $\tilde{y}$,

$w_j^{[l]}[1:n_{l-1}],\ l \in 1:L,\ j \in 1:n_l$ - вектор весовых коэффициентов j-го нейрона в слое l,

$W^{[l]} = W^{[l]}[1:n_l, 1:n_{l-1}] = (w^{[1]}, w^{[2]},\ldots,w^{[L]})^T$,

$b^{[l]} = b^{[l]}[1:n_l]$ - вектор смещений в слое $l \in 0:L$,

$z^{[l]} = z^{[l]}[1:n_l]$ - вектор значений выходов (сумматоров) нейронов слоя $l \in 0:L$,

$g^{[l]}$ - функция активации l - го слоя.

Тогда значение выходов нейронов l-го слоя вычисляется по формулам:

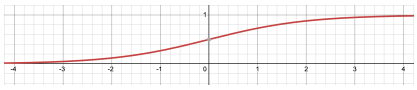
$$a^{[l]}[j] = g(z[j]),$$
$$z^{[l]}[j] = (w_j^{[l]})^T a^{[l-i]} + b^{[l]}[j],\ l \in 1:L,\ j \in 1:n_l,$$

или в векторной форме

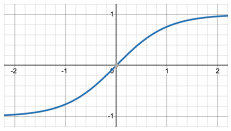
$$a^{[l]} = g(W^{[l]} a^{[l-1]} + b^{[l]}),\ l \in 1:L.$$

В качестве функции активации g используют различные функции. Наиболее часто применимыми являются:

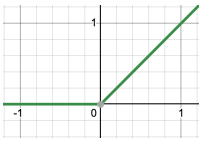
- Сигмоида, $\sigma(x) = \dfrac{1}{1+e^{-x}}$,



- Гиперболический тангенс, $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$,



- Функция RELU (Rectified Linear Unit), $g(x) = max\{0,x\}$



Сигмоиду обычно используют для последнего слоя сети в задаче классификации, поскольку сигмоида, кроме других своих замечательных свойств, принимает значение от 0 до 1, которое можно трактовать как значение вероятности принадлежности входного вектора x одному из двух заданных классов. Пусть задано два класса Ω_1, Ω_2 и величина $y = y(x) = \begin{cases} 0,\ x \in \Omega_1 \\ 1,\ x \in \Omega_1 \end{cases}$.

Тогда значение невязки на входном векторе x равно

$$\Phi(W,b) = yln(\tilde{y}) + (1-y)ln(1-\tilde{y}) \to \min_{W,b}.$$

Окончательно, для всего набора набора обучающих векторов имеем

$$\Phi(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}ln(\tilde{y}^{(i)}) + (1-y^{(i)})ln(1-\tilde{y}^{(i)}) \to \min_{W,b}.$$

Далее введем матрицы:

$X[1:n,1:m] = (x^{(1)}, x^{(2)}, \ldots, x^{(m)})$ - матрица векторов обучающей выборки,
$Z^{[l]}[1:n,1:m] = (z^{[l](1)}, z^{[l](2)}, \ldots, z^{[l](m)}),\ l \in 1:L,$
$A^{[l]}[1:n,1:m] = (a^{[l](1)}, a^{[l](2)}, \ldots, a^{[l](m)}),\ l \in 0:L,$
$Y[1,1:m] = (y^{(1)}, y^{(2)}, \ldots, y^{(m)}),$
$\tilde{Y}[1,1:m] = (\tilde{y}^{(1)}, \tilde{y}^{(2)}, \ldots, \tilde{y}^{(m)}).$

Следовательно, значение активации l-го слоя вычисляется без применения циклов при помощи двух матричных операций:

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]},$$
$$A^{[l]} = + g^{[l]}(Z^{[l]}),\ l \in 1 : L\,.$$

Последовательное вычисление значений активации слоев от начального слоя к последнему называется “прямым проходом” (“Forward Propagation”).

Направление спуска целевой функции определяется как антиградиент или его некоторая модификация.

Вычисление градиента целевой функции $\nabla\,\Phi(W, b)$ также может быть организовано без применения циклов матричными вычислениями, двигаясь в обратном направлении от выхода к входу. Этот процесс носит название “обратного прохода” (“Back Propagation”), а также “обратное распространение ошибки”.

Введем обозначения, которые применяются в написании программного кода:

$$dZ^{[l]} = \frac{\partial\Phi}{\partial Z^{(l)}},\ \ dA^{[l]} = \frac{\partial\Phi}{\partial A^{(l)}},\ \ dW^{[l]} = \frac{\partial\Phi}{\partial W^{(l)}},\ \ db^{[l]} = \frac{\partial\Phi}{\partial b^{(l)}}.$$

Тогда мы можем записать:

$$dZ^{[L]} = A^{[L]} - Y,\quad dW^{[L]} = \frac{1}{m}dZ^{[L]}A^{[L-1]^T},\ \ db^{[L]} = \frac{1}{m}\sum_{i=1}^{m} dZ^{[L]}[i, *\,],$$

и далее, для $l \in 1 : l-1,$

$$dZ^{[l-1]} = dW^{[l]^T}dZ^{[l]}(\nabla g^{[l]}(Z^{[l-1]}),$$

$$dW^{[l-1]} = \frac{1}{m}dZ^{[l-1]}A^{[l-1]^T},$$

$$db^{[l-1]} = \frac{1}{m}\sum_{i=1}^{m} dZ^{[l-1]}[i, *\,]$$

Здесь $dZ^{[l]}[i, *\,]$ означает i-й столбец матрицы $dZ^{[l]}$, $i \in 1 : m,\ l \in 1 : L.$

Замечание. 1) При вычислении $dZ^{[L]}$ использовано свойство сигмоиды, применяемой в последнем слое.

2) Сумма $db^{[l-1]} = \frac{1}{m}\sum_{i=1}^{m} dZ^{[l-1]}[i, *\,]$ вычисляется в программном модуле как одна матричная операция.

Схематично движение данных во время прямого и обратного прохода изображено на Рис. 6.

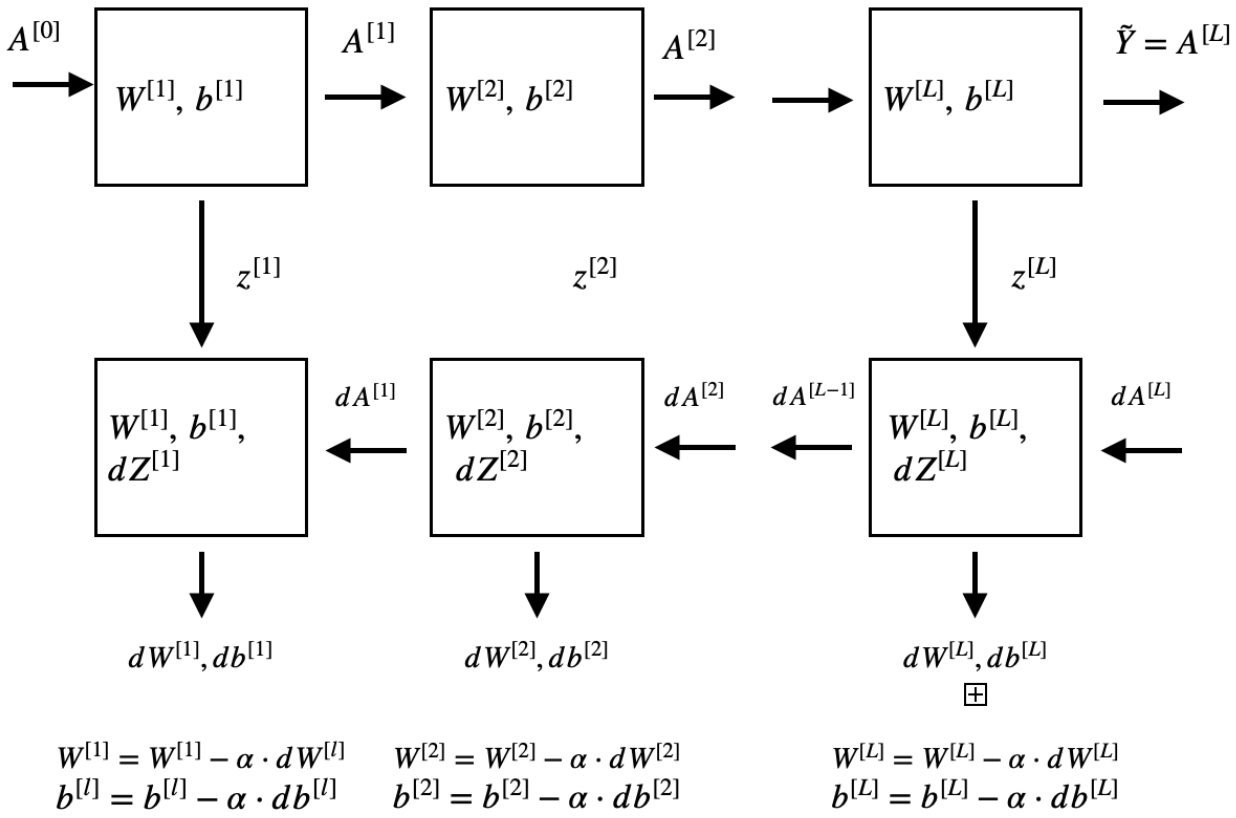


Рис. 6

Сверточные нейронные сети (CNN-Convolutional Neural Network).

Сверточные нейронные сети получили свое название вследствие применения для отдельных слоев операции, подобной операции дискретной свертки двух дискретных сигналов, с одним отличием: фактически вычисляется не свертка, а корреляция с заданным фильтрующим окном. Это позволяет выделить в векторе слоя сети некие признаки. Наибольшее применение сверточные нейронные сети получили в задачах, связанных с обработкой изображений.

Рассмотрим простейший пример свертки, выделяющей вертикальные линии в изображении. На Рис.7 изображена матрица цифрового изображения размеренности 6×6. Меньшему значению числа соответствует более темный серы цвет. Применим к этому изображения сверточный фильтр размеренности 3×3, как показано на рисунке. Фильтрующее окно накладывается на изображение, совпадающие элементы перемножают и складываются, фактически вычисляется скальное произведение фильтра с областью изображения. Затем окно фильтра сдвигается на определенный шаг, в данном

случае на единицу, и производится очередное суммирование. В результате мы получили новое изображение, в котором выделена граница в исходном изображении в виде светлой полосы.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

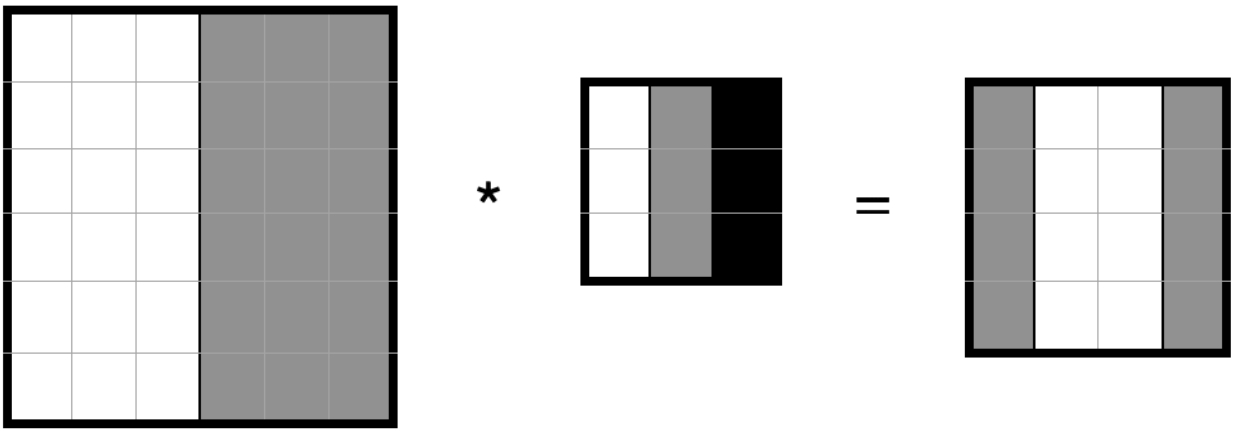


Рис.7

В реальной нейронной сети коэффициенты фильтра могут являться весовыми коэффициентами, которые подлежат обучению. В сверточных нейронных сетях применяются также фильтры, вычисляющие максимальное значение в плавающем окне (Max pool) или среднее значение (Average pool).

Как можно заметить, выходное окно на Рис.7 имеет меньший размер. Для того, чтобы размер изображения не изменялся, применяется прием искусственного расширения исходного изображения некоторым фиксированным значением (padding). В случае, когда фильтрация сохраняет размер изображения, на схемах применяется обозначение “same” (такой же).

На Рис. 8 и Рис. 9 изображены принципиальные схемы двух широко известных сверточных нейронных сетей LeNet-5 и AlexNet соответственно.

Нейронная сеть LeNet-5 была предложена в 1998 году Яном Ле Куном (Yann LeCun) сотрудником AT&T Bell Laboratories для распознавания рукописного текста.

Входное изображение для сети LeNet-5 имеет размер $32 \times 32 \times 1$, к нему применяется шесть фильтров размера 5×5 с шагом $s = 1$, получается 6 изображений размера 28×28. На следующем шаге применяется усреднение (average pool) размера 2×2 с шагом $s = 2$ и т.д. В последних слоях значения активации развертываются в полносвязный слои (FC, Fully Connected) и на выходе - обобщенная логистическая функция (softmax) размеренности 10,

$$\hat{y} = \sigma(y)[k] = \frac{e^{y[k]}}{\sum\limits_{j \in 1:10} e^{y[j]}}, \; k \in 1:10.$$

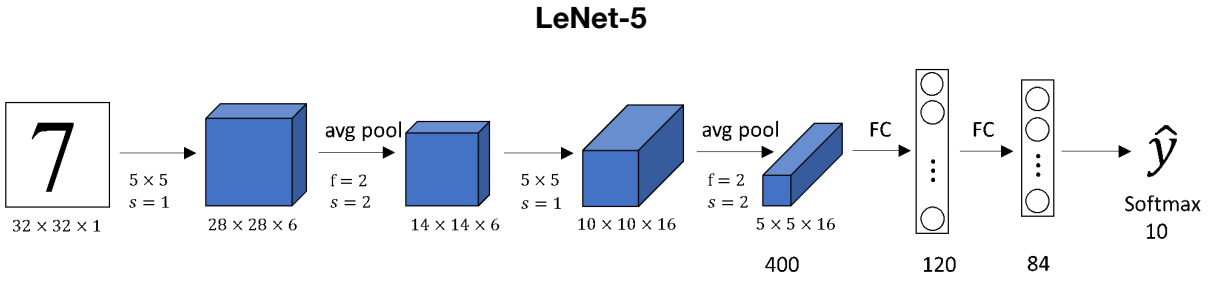


Рис. 8

Нейронная сеть AlexNet, Рис разработана в 2011 году Алексом Крижевски (Alex Krizhevsky) в сотрудничестве с Ильей Суцкевером (Ilya Sutskever) и Джефри Хинтоном (Geoffrey Hinton). Последний был научным руководителем Алекса Крижевского.

У сверточных сетей есть еще одно важное преимущество по сравнению с полносвязными нейронными сетями, они имеют существенно меньшее количество параметров, по которым проводится обучение (оптимизация). В таблице 2 приведены параметры сети LeNet-5.

Слой	**Размеренность вектора активации**	**Количество элементов активации**	**Количество параметров**
Вход	$32 \times 32 \times 3$	3,072	0
Свертка (f=5, s=1)	$28 \times 28 \times 6$	4,704	156
Усреднение (f=2, s=2)	$14 \times 14 \times 6$	1,176	0
Свертка (f=5, s=1)	$10 \times 10 \times 16$	1,600	416

Усреднение (f=2, s=2)	$5 \times 5 \times 16$	400	0
Полносвязный	120×1	120	48,001
Полносвязный	84×1	84	10,081
Выход (Softmax)	10×1	10	841

Таблица 2.

Легко заметить, как увеличивается количество обучаемых параметров при применении полносвязный слоев.

AlexNet

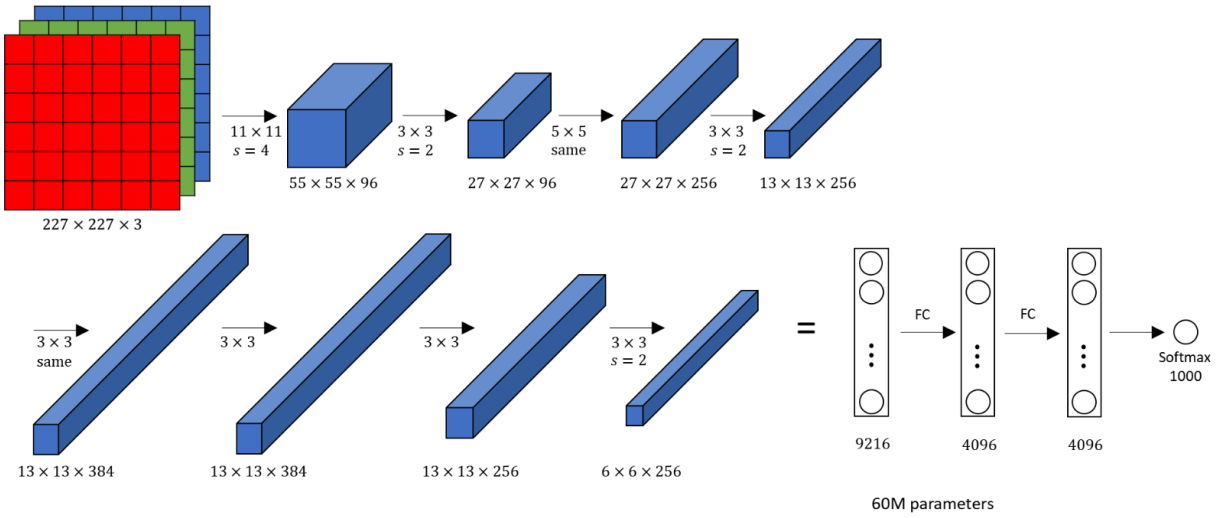


Рис. 9

2. Выбор направления спуска

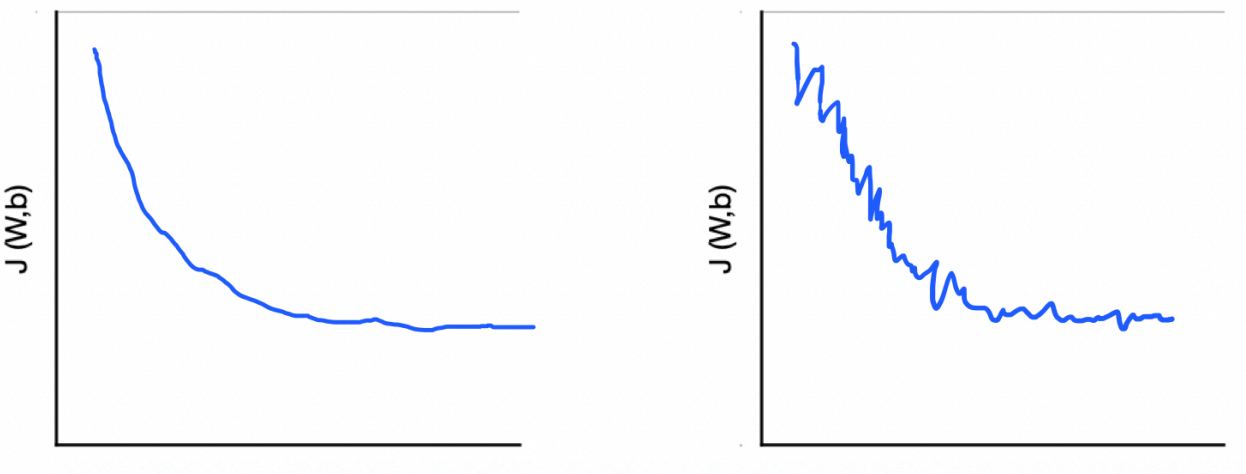
Одним из ключевых факторов, влияющих на сходимость метода оптимизации и на ее скорость являются выбор направления спуска целевой функции и регулировка шага. В практике обучения нейронных сетей обычно применяют либо фиксированный шаг, либо шаг, уменьшающийся в процессе приближения к локальному минимуму по некоторому простому эмпирическому закону, подобранному в процессе обучения конкретной модели.

В методах выбора направления спуска выделим наиболее часто встречающиеся.

В первой группе - методы, вносящие элемент случайности в выбор направления спуска:

- Метод минипакетов,
- Стохастический градиентный спуск.

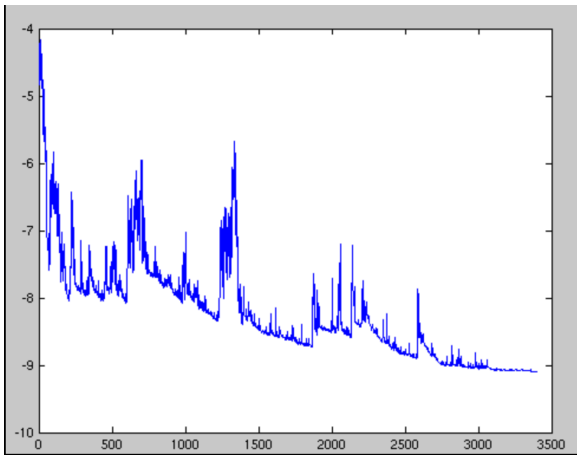
Градиентный спуск **Метод минипакетов**



Количество итераций Количество итераций

Рис. 10

Стохастический градиентный спуск



Количество итераций

Рис.11

Вторая группа методов основана на некотором экспонециальном сглаживании градиента. К эти методам относятся методы:

- RMSprop (Root Mean Square prop) (квадратичное сглаживание),

- Adam (метод Адама),

В таблице 3 приведено краткое описание выше названных алгоритмов.

	Название метода	Краткое описание
1	Minibatch (метод минипакетов)	Обучающая выборка разбивается на относительно небольшие подборки, минипакеты (minibatch) и для каждого минипакета выполняется один шаг градиентного спуска (прямой и обратный проход. Один цикл по всем подборкам называется “эпохой” (“epoch”).
2	SGB (Stochastic Gradient Descent) (стохастический градиентный спуск)	Обучающая выборка случайным образом перемешивается и для каждого элемента (входного вектора) осуществляется ода итерация градиентного спуска. Является предельным случаем метода минипакетов.
3	RMSprop (Root Mean Square prop) (квадратичное сглаживание)	$W = W - \alpha \cdot \frac{dW}{\sqrt{S_{dW}}}, \;\; b = b - \alpha \cdot \frac{db}{\sqrt{S_{db}}},$ где $S_{dW} = \beta \cdot S_{dW} + (1-\beta) \cdot dW^2, \;\; S_{db} = \beta \cdot S_{db} + (1-\beta) \cdot db^2,$ здесь возведение в степень dW^2 и db^2 - поэлементное, β - параметр метода.
4	Adam (метод Адама)	$W = W - \alpha \cdot \frac{v^c_{dW}}{\sqrt{S^c_{dW}}}, \;\; b = b - \alpha \cdot \frac{v^c_{db}}{\sqrt{S^c_{db}}},$ $V^c_{dW} = \frac{V_{dW}}{1-\beta_1^t}, \; V^c_{db} = \frac{V_{db}}{1-\beta_1^t}, \; S^c_{dW} = \frac{S_{dW}}{1-\beta_2^t}, \;\; S^c_{db} = \frac{S_{db}}{1-\beta_2^t},$ где t - номер итерации, $V_{dW} = \beta_1 \cdot V_{dW} + (1-\beta_1) \cdot dW, \;\; V_{db} = \beta_1 \cdot V_{db} + (1-\beta_1) \cdot db,$ $S_{dW} = \beta_2 \cdot S_{dW} + (1-\beta_2) \cdot dW^2, \;\; S_{db} = \beta_2 \cdot S_{db} + (1-\beta_2) \cdot db^2,$ β_1, β_2 - параметры метода.

5	SGB (Stochastic Gradient Descent) (стохастический градиентный спуск)	Обучающая выборка случайным образом перемешивается и для каждого элемента (входного вектора) осуществляется ода итерация градиентного спуска. Является предельным случаем метода минипакетов.

Таблица 3.

3. Регуляризация

Методы регуляризации обучения нейронной сети можно разделить на группы методов:

1) Изменение целевой функции путем добавления слагаемого, содержащее норму вектора параметров. Это прежде всего регуляризация Тихонова (квадратичная) и L_1 регуляризация.
2) Уменьшение размеренности задачи и добавление случайного возмущения. Очень простым и эффективным является метод прореживания. Он заключается в том, что на определенном шаге из вычислительной схемы просто исключаются случайным образом выбранные узлы и шаг градиентного спуска осуществляется без них. На следующем шаге исключаются другие узлы.
3) “Улучшение” обучающей выборки. К этим методам можно отнести:
   - Увеличение обучающей выборки путем ее трансформации (Data augmentation), Рис.12,
   - Нормализация входов и значений активаций.

К отдельному методу регуляризации можно отнести “Раннюю остановку”. Краткое описание методов регуляризации приведено в Таблице 4.

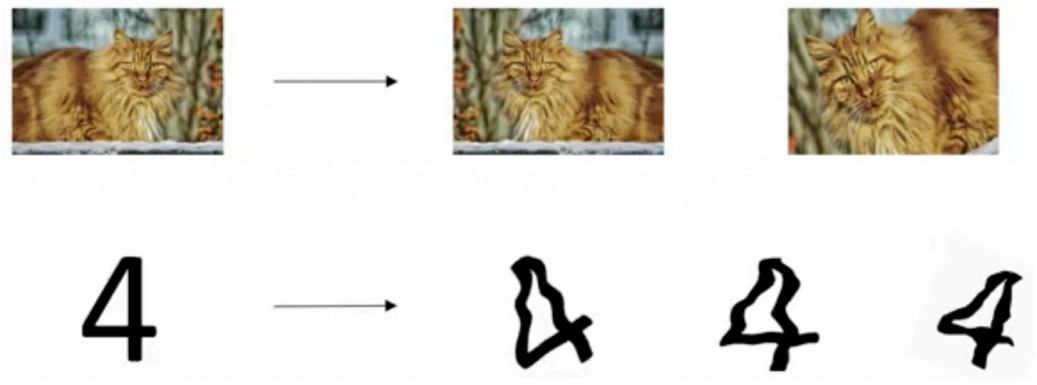


Рис. 12

	Регуляризация	Краткое описание
1	Квадратичная	$\Phi(W,b) = -\frac{1}{m}\sum_{1}^{m}(y\ln(\hat{y}) + (1-y)\ln(1-\hat{y})) + \frac{1}{2m}\|W\|_F^2$
2	L_1	$\Phi(W,b) = -\frac{1}{m}\sum_{1}^{m}(y\ln(\hat{y}) + (1-y)\ln(1-\hat{y})) + \frac{1}{m}\|W\|_1$
3	Прореживание (Dropout)	В каждом слое в процессе обучения случайным образом “удаляется” некоторое количество нейронов.
4	Увеличение обучающей выборки (Data augmentation)	В обучающую выборку добавляются данные, полученные из исходных путем различных трансформаций, например, поворот изображения, растяжение, отражение и пр.
5	Ранняя остановка	Процесс оптимизации обрывается в зависимости от скорости убывания целевой функции до достижения стационарной точки. Помогает избежать “переобучения”.
6	Нормализация входов	Для каждой партии (minibatch) производится нормализация (по среднему и дисперсии), причем не только внешнего входа, но и входов для отдельных слоев.

Таблица 4.

Заключение.

В настоящем докладе очень кратко рассмотрены только несколько ключевых вычислительных аспектов нейронных сетей.

1. Векторизация означает приведение вычислений к матричному виду, который позволяет эффективно использовать современные программные средства и современную архитектуру вычислительных процессоров. Векторизация применяется во всех типах современных нейронных сетей как для прямого вычисления функционала ошибки, так и для вычисления градиента при помощи обратного прохода.
2. В подавляющем большинстве случаев для поиска весовых коэффициентов сети (обучения) используется градиентый спуск и его модификации.
3. Борьба с неустойчивостью вычислений, регуляризация, осуществляется путем модификации функционала ошибок традиционными способами, например при помощи квадратной добавки, а также путем обработки обучающей выборки и другими эмпирическим способами.