Санкт-Петербургский государственный университет
Математико-механическии факультет
Кафедра исследование операций

Зенков Дмитрий Антонович

Методы машинного обучения в задаче распознавания изображений

Выпускная квалификационная работа бакалавра

Научный руководитель:
к.ф.-м.н., доцент М. С. Ананьевский

Рецензент:
д.т.н., профессор И. Б. Фуртат

Санкт-Петербург
2019

Saint Petersburg State University
Faculty of Mathematics and Mechanics
Chair of Operation Research

Zenkov Dmitry Antonovich

Machine learning methods in the problem of image recognition

Graduation Thesis

Scientific Supervisor:
M. S. Ananyevskiy

Reviewer:
I. B. Furtat

Saint Petersburg
2019

Оглавление

Введение

Рассматривалась общая задача распознавания рукописного текста, апробация происходила на базе данных рукописных цифр MNIST[1]. MNIST представляется из себя переработку оригинального набора чёрно-белых образцов NIST с добавленными образцами написанными студентами.

В статье [1] В. А. Якубовича, рассматривалась задача распознавания рукописных цифр. Современный подход решения данной задачи основанный на сверточных нейронных сетях базируется на статье [8] Yann Lecun и др. Я решил задачу распознавания рукописных цифр с помощью свёрточной нейронной сети.

Преимущества такой модели:

- свёрточные нейронные сети хорошо зарекомендовали себя в задачах по распознаванию изображений;
- уменьшение количества обучаемых параметров и повышение скорости обучения по сравнению с полносвязной нейронной сетью;

В работе рассказано с помощью каких операций производится обучение свёрточной нейронной сети и показывается дальнейшее распознавание ею объектов из MNIST.

[1] `http://yann.lecun.com/exdb/mnist/`

1. Математическая постановка задачи

Рассмотрим задачу классификации с учителем.

Ω — множество объектов распознавания.

$\omega \in \Omega$ — объект распознавания.

$g\colon \Omega \to M$, $M = \{1, 2, \ldots, m\}$ — индикаторная функция, разбивающая пространство образов на m непересекающихся классов $\Omega^1, \Omega^2, \ldots, \Omega^m$.

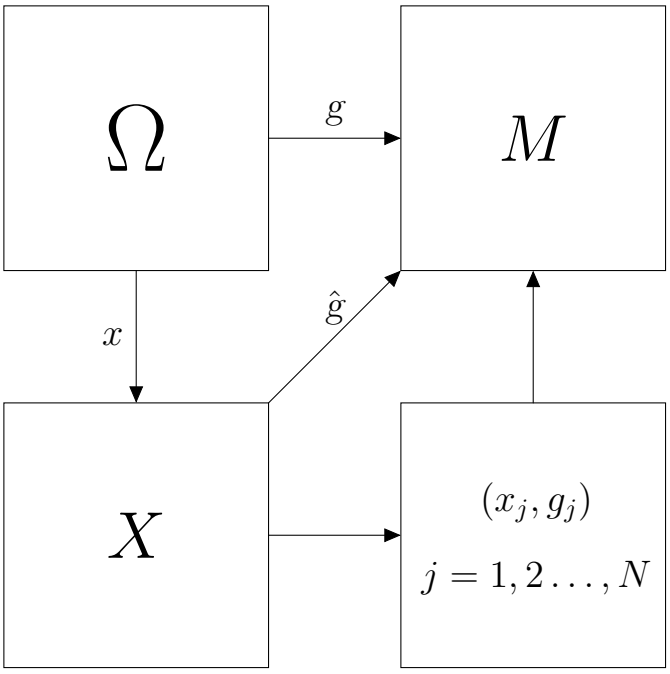
X — пространство наблюдений (признаки).

$x\colon \Omega \to X$ — функция, ставящая в соответствие каждому объекту ω точку $x(\omega)$ в пространстве наблюдений. Вектор $x(\omega)$ — это образ объекта, воспринимаемый наблюдателем.

$\hat{g}\colon X \to M$ – решающее правило (оценка для $g(\omega)$ на основании $x(\omega)$).

Пусть $x_j = x(\omega_j)$, $g_j = g(\omega_j)$, $j = 1, 2 \ldots, N$ — доступная информация о функциях $g(\omega)$ и $x(\omega)$, сами функции неизвестны. Тогда (g_j, x_j), $j = 1, 2 \ldots, N$ — множество прецендентов.

Задача: построить решающее правило $\hat{g}(x)$, чтобы распознавание проводилось с минимальным числом ошибок на заданном множестве.



2. Постановка задачи на естественном языке

Ставилась задача распознавания рукописных цифр от 0 до 9. Для апробации был использован стандартный набор данных MNIST (42000 изображений обучающей выборки, 28000 изображений тестовой выборки).

Программе представлялась картинка с изображением одной из 10 цифр 0, 1,. . . , 9. Программа должна была правильно распознать, что за цифра нарисована на картинке.

3. Техническая постановка задачи

Имеем:

- обучающая выборка: размечанные изображения;
- тестовая выборка: размечанные изображения, но разметка используется только для проверки классификатора.

Изображение задавалось 28×28 пикселями, в 256 оттенках серого цвета (цветовой режим Greyscale, значение 0 представляет чёрный цвет, а значение 255 — белый).

Цель: распознать рукописную цифру на тестовых изображениях (выбрать правильную метку от 0 до 9).

Метрика качества: Процент правильно распознанных цифр (Accuracy).

Продемонстрированы примеры объектов из MNIST (рис. 1). А также показано распределение объектов обучающей выборки (рис. 2).

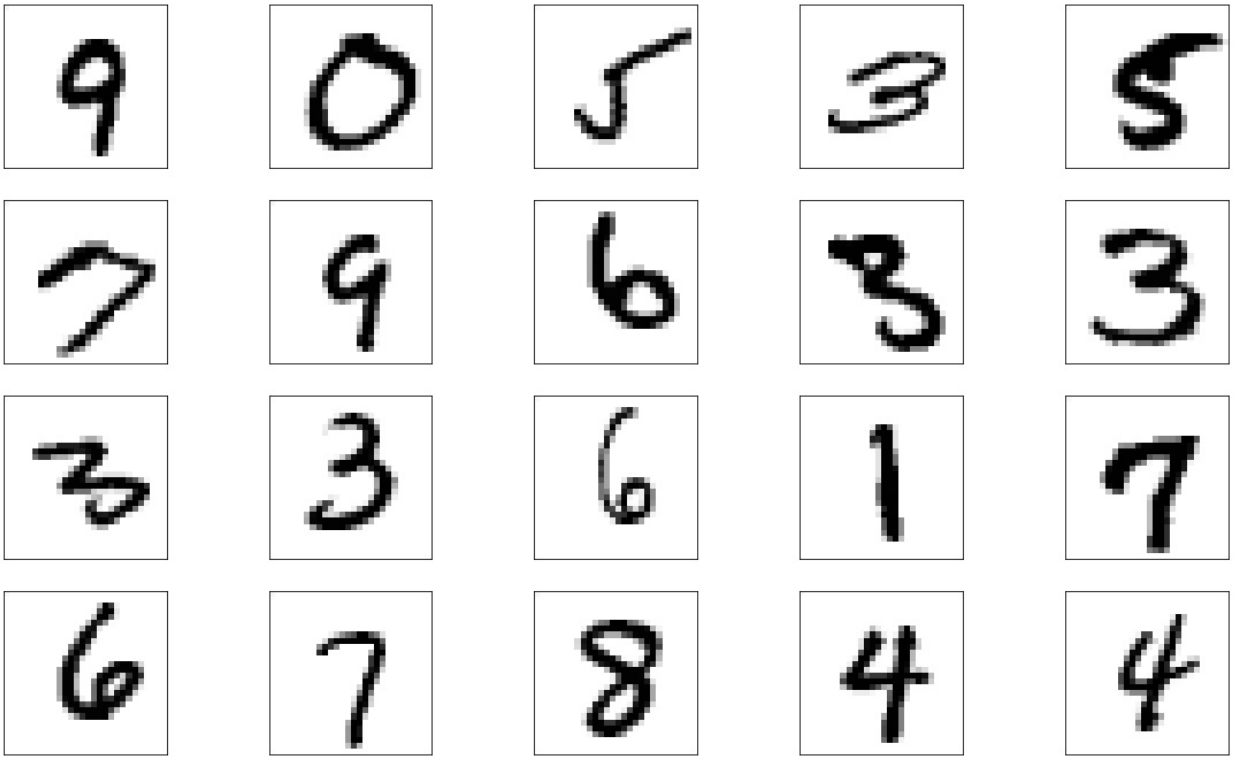


Рис. 1. Пример объектов из набора данных MNIST

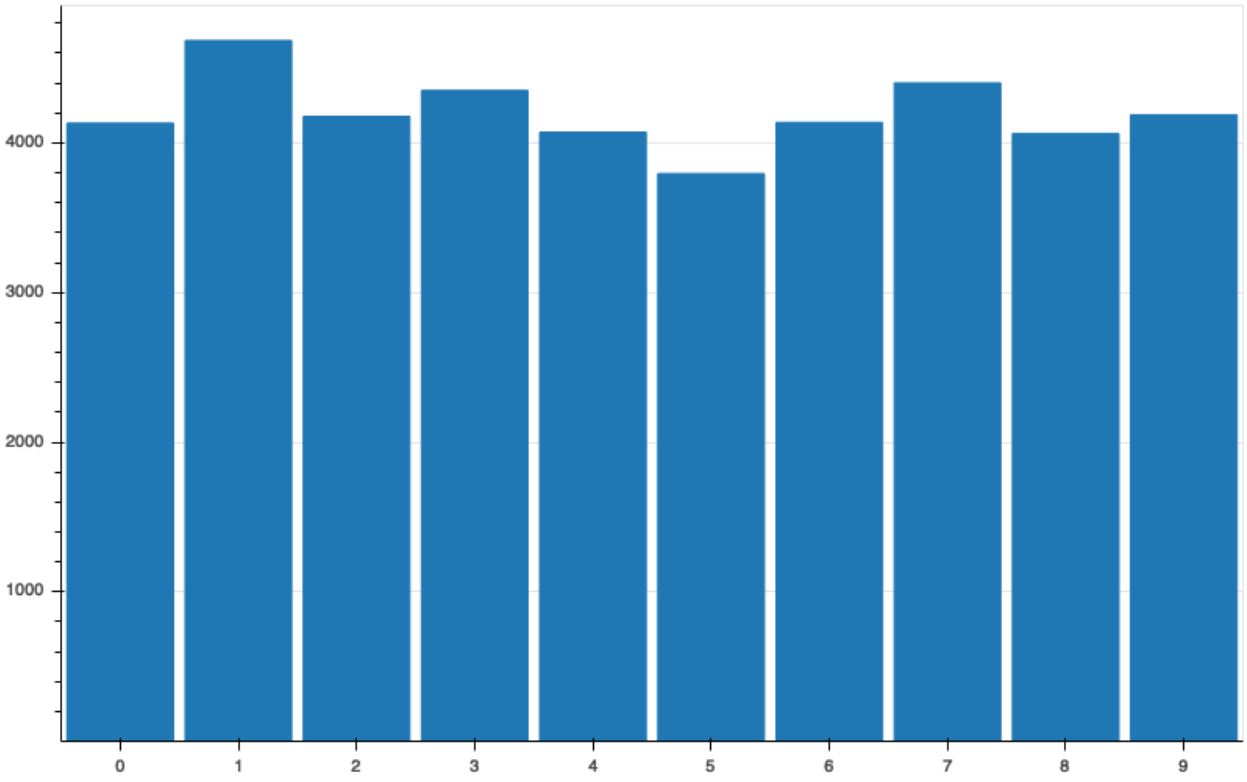


Рис. 2. Распределение цифр в обучающей выборке

4. Архитектура сети

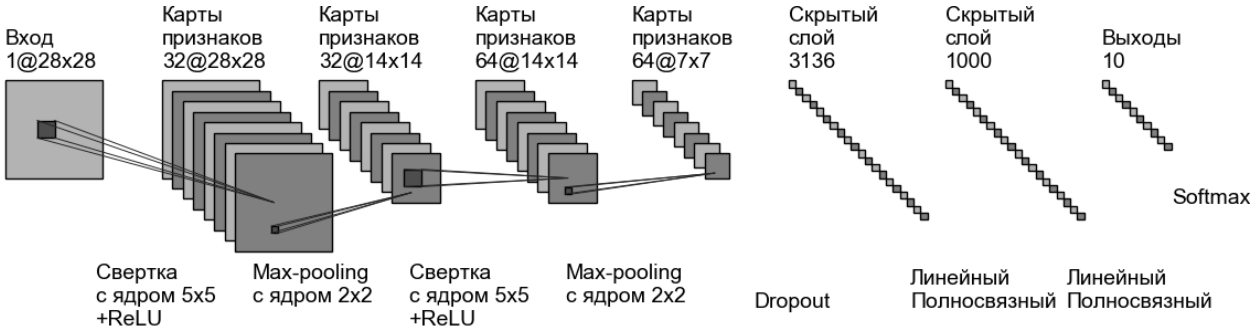


Рис. 3. Используемая архитектура нейронной сети

1. Сверточный слой:
   - 32 карты признаков;
   - ядро 5×5, шаг ядра 1;
   - значение padding 2×2;
   - активационная функция ReLU;
   - max-pooling ядро 2×2, шаг ядра 1.
2. Сверточный слой:
   - 64 карты признаков;
   - ядро 5×5, шаг ядра 1;
   - значение padding 2×2;
   - активационная функция ReLU;
   - max-pooling ядро 2×2, шаг ядра 1.
3. Dropout;
4. Полносвязный слой (входов $64 \times 7 \times 7$, выходов 1000);

5. Полносвязный слой (входов 1000, выходов 10), активационная функция Softmax.

Свёрточные нейронные сети показывают хороший результат в работе с изображениями. В их основе лежит идея выделения признаков исходного изображения. Такое выделение признаков позволяют делать операции свертки и субдискретизации (pooling). Также я применил dropout регуляризацию, чтобы увеличить способность нейронной сети выделять признаки.

На (рис. 3) схематично изображена используемая архитектура свёрточной нейронной сети. Получившиеся количество настраиваемых параметров нейронной сети: 3199106.

4.1. Свертка

Свертка (взаимнокорреляционная функция) — основная операция свёрточной нейронной сети, позволяющая выделять признаки изображения. Свертку задается формулой:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n),$$

где K — двумерное ядро, I — двумерное изображение.

Ядро перемещается над двумерным изображением, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, затем суммирует все полученные значения в один выходной пиксель. Таких ядер в одном слое сети может быть несколько (настраиваемый пользователем параметр) и зависит от того, сколько признаков в изображении пользователь хочет выделить. Выход такой функции называется картой признака.

Количество настраиваемых параметров у свертки заметно меньше, чем на полносвязном слое. Если обозначить $futmap_{in}$ — количество карт

признаков подающихся на вход, kH, kW — высота и ширина ядра соответственно, а $futmap_{out}$ — количество карт признаков на выходе. Тогда количество параметров свертки можно вычислить по формуле:

$$number\ of\ parameters = kH \times kW \times futmap_{in} \times futmap_{out}$$

Также представлена визуализация свертки (рис. 4).

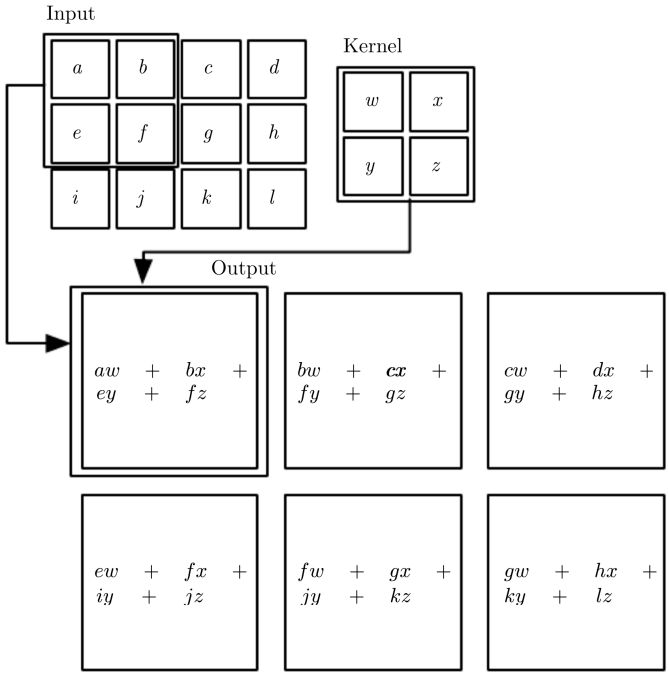


Рис. 4. Операция свертки с ядром 2×2

4.2. Субдискретизация

Суть субдискретизации заключается в нелинейном уплотнение карты признаков, группа пикселей уплотняется до одного пикселя, проходя нелинейное преобразование. Такое преобразование уменьшает количество параметров нейронной сети в дальнейшем, оставляя при этом выделенные признаки.

Мною была выбрана операция субдискретизации с функцией максимума (max-pooling), как наиболее предпочтительная из статьи [6]. Формула

субдискретизации с функцией максимума:

$$out(h,w) = \max_{m=0,\ldots,kH-1} \max_{n=0,\ldots,kW-1} input(stride_h \times h + m, stride_w \times w + n),$$

где kH, kW — высота и ширина ядра соответственно, $stride_h, stride_w$ — размер шага фильтра по высоте и ширине соответственно, $input$, out — изображение на входе и на выходе соответственно.

4.3. Dropout регуляризация

Dropout регуляризация избавляет от взаимоадаптации нейронов во время обучения нейронной сети.

Во время обучения производится выключение (dropping out) нейронов с вероятностью p, таким образом, а вероятность, что нейрон останется включенным, составляет $1-p$. Выключение нейрона означает, что при любых входных данных или параметрах он возвращает 0.

На каждом шаге обучения генерируется вектор случайных величин $D = (D_1, \ldots, D_{n_1})$ распределенный по закону Бернулли, где n_1 — количество нейронов в слое, к которому применяется dropout. Этот вектор показывает какие нейроны выключатся, а какие останутся включенными на данном шаге обучения.

Применение dropout к i-ому нейрону:

$$O_i = D_i f\left(\sum_{k=1}^{n_0} w_k x_k + b\right) = \begin{cases} f(\sum_{k=1}^{n_0} w_k x_k + b), & \text{если } D_i = 1 \\ 0, & \text{если } D_i = 0 \end{cases},$$

где w_k, b — веса выходов нейронов предыдущего слоя и смещение, n_0 — количество нейронов предыдущего слоя, f — функция активации, O_i — выходное значение i-го нейрона, $P(D_i = 0) = p$.

Когда сеть уже обучена, то выход на i-ом нейроне выглядит так:

$$O_i = (1-p)f\left(\sum_{k=1}^{n_0} w_k x_k + b\right).$$

5. Алгоритм обучения

В данном разделе рассматривается процесс обучения данной свёрточной нейронной сети. Определяется функцию, которую нужно минимизировать. По какому алгоритму обновляются веса нейронной сети. Каким методом считается градиент функции. Вводится определения используемых функция активации.

Назовем функционалом качества такую функцию:

$$f(X, \boldsymbol{\theta}) = -\frac{1}{|X|} \sum_{x \in X} \sum_{i=0}^{9} t_i(x) \log(softmax(\boldsymbol{z}(x, \boldsymbol{\theta}))_i),$$

где X — изображения, которые подаем в нейронную сеть,
$\boldsymbol{t}(x)$ — единичный орт размерности 10, обозначающий класс объекта x,
$\boldsymbol{z}(x)$ — выходной вектор нейронной сети при подаче объекта x,
$\boldsymbol{\theta}$ — параметры нейронной сети (в рассматриваемом случае $\boldsymbol{\theta} \in \mathbb{R}^{3199106}$).

Такой функционал принимает значения в области $(0, \infty)$. $f(X, \boldsymbol{\theta}) \approx 0$, когда $softmax(\boldsymbol{z}(\boldsymbol{x}))_i \approx 1$ для каждого $x \in X$. Это значит, что нейронная сеть выдает меру принадлежности каждого объекта к правильной метке значительно больше, чем к другим.

5.1. Функции активации

Главной задачей функции активации внести нелинейность в нейронную сеть.

Функция активации ReLU (Rectified Linear Unit) начала использоваться относительно недавно и приобрела широкое распространение. Формула ReLU:

$$ReLU(x) = \max(0, x),$$

где x — входной сигнал в нейрон.

К преимуществом такой функции активации можно отнести:

- простое вычисление;
- не происходит насыщение функции.

Единственный недостаток, что при большом градиенте функции нейрон может больше никогда не обновляться, но это проблема решается посредством выбора надлежащей скорости обучения.

Функция активации softmax используется на последнем слое нейронной сети. Представляет собой обобщение логистической функции для большей размерности. Размерность вектора преобразованного функцией остается той же. Компоненты вектора принимают значения в интервале $(0, 1)$. Сумма компонент вектора равна 1.

$$softmax(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Выходы функции можно интерпретировать как меру принадлежности к каждому классу.

5.2. Adam (Adaptive Moment Estimation)

Главная задача нейронной сети минимизировать функционал качества. Функционал качества зависит от объекта, который подали на вход в нейронную сеть, x и от параметров нейронной сети $\boldsymbol{\theta}$. Минимизация функционала качества $\left(\underset{\theta}{\operatorname{argmin}} f(\theta)\right)$ производится за счет автоматического изменения весов в нейронной сети по такому алгоритму:
Значения θ задаются из непрерывного равномерного распределения.

$$
\begin{aligned}
t &= t + 1 \\
m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \nabla f_t(\theta_{t-1}) \\
\upsilon_t &= \beta_2 \upsilon_{t-1} + (1 - \beta_2) \left(\nabla f_t(\theta_{t-1})\right)^2 \\
\widehat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\widehat{\upsilon}_t &= \frac{\upsilon_t}{1 - \beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{\widehat{\upsilon}_t} + \varepsilon} \widehat{m}_t
\end{aligned}
$$

С параметрами: $\alpha = 0.0005$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $m_0 = 0$, $\upsilon_0 = 0$, $t = 0$.

Все параметры, кроме α — скорость обучения, были взяты как оптимальные параметры найденые в ходе экспериментов из статьи [3]. Параметр α был взят с учетом данной задачи.

Подсчет градиента функции производится с помощью метода обратного распространения ошибки. Он поочередно от последнего слоя к первому вычисляет градиент функционала качества по параметрам текущего слоя. Это вычисление возможно так как, нейронная сеть представляет собой композицию функций, для которой можно использовать правило дифференцирования сложной функции.

6. Обучение свёрточной нейронной сети

Нейронная сеть была написана на python 3.6 с использованием библиотеки pytorch. Из-за большого количества данных, не помещающихся в оперативную память компьютера, в нейронную сеть загружались за раз 100 изображений из обучающей выборки. Все изображения прошли 40 раз через свёрточную нейронную сеть.

Выбор метки для объекта прошедшего нейронную сеть, осуществлялся выбором номера наибольшей компоненты выходного вектора нейронной сети.

7. Заключение

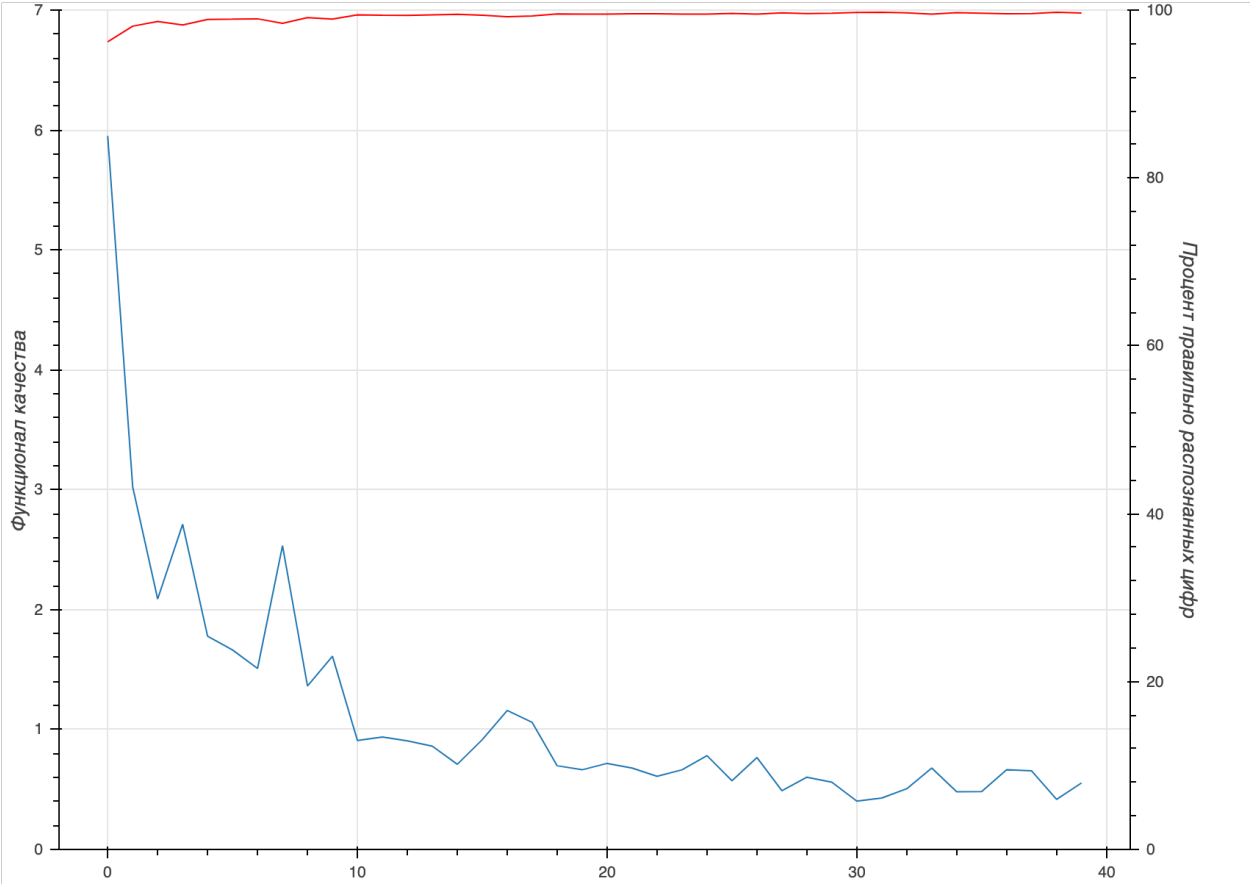


Рис. 5. Процесс обучения нейронной сети

На (рис. 5) изображены графики функционала качества и процента правильно распознанных цифр из обучающей выборки зависящие от того, сколько раз вся обучающая выборка прошла через нейронную сеть. Из графиков видно, что процент правильно распознанных цифр после того, как обучающая выборка прошла через нейронную сеть 10 раз был больше 99%.

На тестовом наборе данных процент правильно распознанных цифр составляет 99.385%. Получившаяся нейронная сеть попала в топ 25% на соревновании kaggle Digit Recognizer, из более чем 3000 участников.

Литература

[1] Якубович В. А. “Машины, обучающиеся распознаванию образов”. В: *Методы вычислений* 2 (1963), с. 95—131.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[3] Diederik P. Kingma и Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: `1412.6980 [cs.LG]`.

[4] Ian Goodfellow, Yoshua Bengio и Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[5] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[6] Dominik Scherer, Andreas C. Müller и Sven Behnke. “Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition.” В: *ICANN (3)*. Lecture Notes in Computer Science. Springer, 2010, с. 92—101.

[7] Geoffrey E. Hinton и др. “Improving neural networks by preventing co-adaptation of feature detectors”. В: *CoRR* abs/1207.0580 (2012).

[8] Yann Lecun и др. “Gradient-Based Learning Applied to Document Recognition”. В: *Proceedings of the IEEE* 86 (1998), с. 2278—2324.

[9] http://www.machinelearning.ru/wiki/images/1/1e/Sem07_ann.pdf.

[10] http://cs231n.github.io/convolutional-networks/.