Отчет по проведенной НИРС

В качестве предметной области была предложена задача распознавания модели автомобиля по фотографии и, хотя на данный момент уже существуют работающие решения, до сих пор остается не изученной проблема дообучения таких систем. Базовый подход заключается в применении методов машинного обучения (ML — Machine Learning), таких как нейронные сети (NN — Neural Network, или ANN — Artificial Neural Network) и деревья принятия решений (decision tree).

Машинное обучение можно определить как математическую дисциплину, или набор математических, статистических, численных методов и различных подходов из теории вероятностей для решения задачи автоматического выделения информации (information extraction) и ее интеллектуального анализа (data mining) из неструктурированного набора входных данных. Более подробно — ставится задача сопоставления заданному значению (объекту) некоторого действия (ответа), таким образом можно говорить об обобщении задачи аппроксимации функции.

Машинное обучение активно используется при классификации данных, например, распределение новостных статей по рубрикам. Отдельное внимание уделяется обработке изображений, их автоматическому аннотированию [1] или выделению одного определенного объекта на изображении, задача распознавания рукописного текста (OCR — Optical Character Recognition).

Основные подходы машинного обучения при работе с изображениями — это использование нейронных сетей и деревьев принятия решений. Перед использованием построенной модели ее необходимо обучить на тестовом наборе данных, для нейронных сетей выработать базу признаков. Алгоритмы обучения решают задачу оптимизации, занимаются подбором неизвестных коэффициентов в математической модели описывающей конкретный метод машинного обучения. Процесс обучения требует большого количества вычислительных операций, что заставляет искать альтернативные подходы при работе с предметной областью, в которую могут добавляться новые классы объектов. Вместо полного переобучения модели используют алгоритмы частичной корректировки коэффициентов (fine tuning). Такой подход дообучения требует меньше времени, но связан с потерей качества, так как не добавляет в модель признаков объектов новых классов.

Деревья принятия решений позволяют достаточно легко и с хорошей точностью решать задачу классификации по заданному набору предикатов. Однако, необходимо отметить, что ее дообучение и добавление новых классов требует перестроения существующего дерева. Таким образом, применение этого подхода в рассматриваемой предметной области остается под вопросом, хотя при использовании деревьев и получается добиться хороших результатов распознавания.

Использование полносвязных нейронных сетей обратного распространения для решения этой задачи почти не осуществимо на практике, так как требует огромного числа вычислительных ресурсов. Вместо них используются нейросети специальной архитектуры, нацеленной на обработку и классификацию изображений. Существуют готовые решения, предоставляющие возможность развертывания таких сетей, а также алгоритмы дооубечения, позволяющие добавлять новые классы без полного перестроения выработанной модели. Но до сих пор остается открытым вопрос — как долго можно осуществлять дообучение без существенной потери качества классификатора.

В данном отчете содержится описание рассмотренных алгоритмов машинного обучения и их применение для задач классификации (ирисы Фишера для деревьев принятия решений, распознавание рукописных цифр для сверточных нейронных сетей и метод Виолы-Джонса нахождения лиц на фотографиях).

Разделы

1 Деревья принятия решений

Дерево принятия решений является графом, в узлах которого записаны условия определяющие переход в следующую вершину, а в листьях хранятся значения целевой функции (рисунок 1). Таким образом, решение задачи сводится к спуску из корня дерева до его листа.

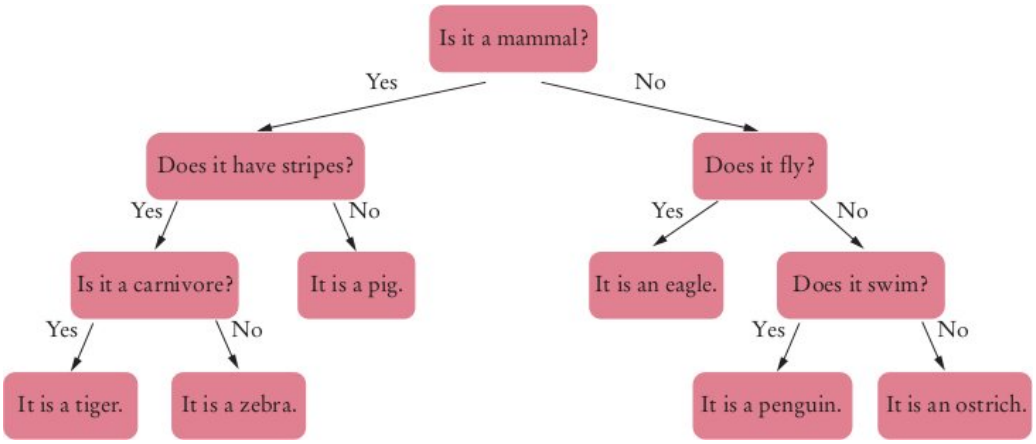


Рисунок 1 – Пример дерева решений классификации животных

В зависимости от решаемой задачи в листьях дерева могут храниться значение одного из двух основных типов: либо это конкретный класс рассматриваемой предметной области при решении задачи классификации, либо число (регрессионная модель), определяющее вероятность наступления некоторого события (победа команды в предстоящем матче) или прогнозируемую величину исследуемой функции (цена от продажи квартиры). В качестве входных данных дерево решений принимает массив параметров с опорной информацией, используемой в узлах дерева для определения перехода, например, для расчета вероятности выигрыша футбольной команды это будут: информация о положении в турнирной таблице, погоде и предыдущих играх.

1.1 Общий подход формирования дерева

Деревья решений имеют очень простую и наглядную структуру, что позволяет формировать их вручную для небольшого набора данных. Тем не менее, существуют разные схемы для автоматического обучения и выращивания деревьев. Каждый из таких методов имеет свои преимущества и недостатки, но один из подходов стал основой большинства других решений, поскольку

оказался очень быстродействующим и простым в реализации. Этот алгоритм известен под названием рекурсивного секционирования [2] и действует по принципу инкрементного построения дерева.

Процесс выращивания начинается с пустого дерева и полного набора данных. Первым этапом решается задача поиска приемлемой точки разбиения исходной выборки на подмножества. С точки зрения реализации алгоритма, применяемый способ выбора атрибута разделения не имеет значения. Таким образом создается первый узел в дереве, разбивающий набор данных на несколько подмножеств, после чего данный процесс выполняется повторно, применительно к каждому из подмножеств, для создания поддеревьев. Критерий прекращения выращивания дерева выбирается в зависимости от решаемой задачи.

Для создания узлов принятия решений необходимо установить какими должны быть критерии секционирования. Как правило, при выборе атрибутов разбиения применяется жадный алгоритм, предусматривающий выбор наилучшего из приемлимых атрибутов.

1.2 Функция оценки

Основная идея при рекурсивном построении дерева — это создание «хорошего» разбиения для текущего узла. Чтобы оценить качество получаемых поддеревьев и выбрать среди возможных вариантов самое оптимальное, вводят специальную функцию оценки.

Для конкретности, рассмотрим пример построения дерева принятия решений задачи классификации. На вход подается тестовая выборка $< x_i, b_i >$, где x_i — i-ый вектор параметров, соответствующий классу b_i. Определено множество предикатов H и каждый не листовой элемент дерева связан с некоторым $h \in H$.

На рисунке 2 показан пример классификатора, разбивающего двумерное пространство точек на четыре части. Множество предикатов задано функциями вида $H = \{x_i > \alpha | \alpha \in \mathbb{R}\}$.

Использование функции оценки при построении дерева должно увеличить точность создаваемого классификатора. Определим ошибку в каждой вершине v как $E(v)$, следовательно суммарную ошибку по всему дереву можно

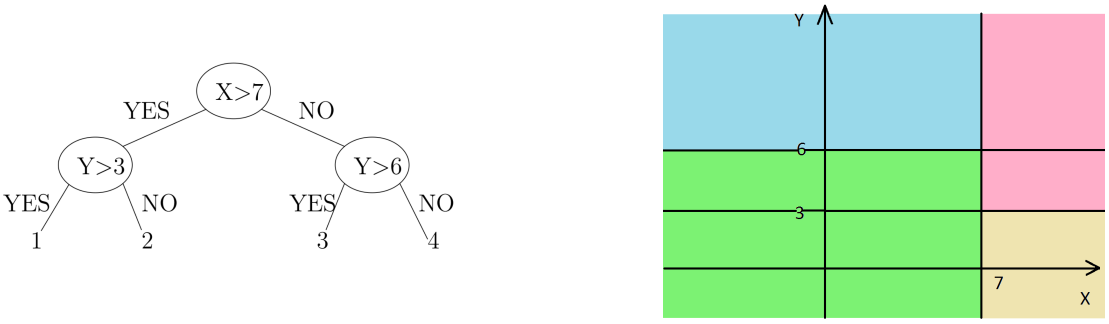


Рисунок 2 – Разбиение двумерного пространства точек деревом принятия решений; класс 1 — розовый цвет, класс 2 — желтый, класс 3 — голубой и класс 4 — зеленый

вычислить как $E(T) = \sum_{v \in T} Prob(v)E(v)$, где T — дерево принятия решений, а $Prob(v)$ — вероятность перехода в вершину v, т. е. если по предикату H из вершины v осуществляется переход в одно из двух поддеревьев с корнями в вершинах v_1 и v_2, вероятности для них можно рассчитать по формулам:

$$Prob(v_1) = P(H)$$

$$Prob(v_2) = 1 - P(H)$$

где $P(H)$ — вероятность перехода из вершины v в v_1. На очередном этапе рекурсивного алгоритма выращивания в качестве функции оценки можно использовать $E(T)$, где T — это дерево с предполагаемым разбиением и корнем в текущей вершине. Однако, такой подход не всегда работает, например, если в исходной вершине v наблюдалась ошибка $E(v) = 0.2$ и один из вариантов ее разделения предполагает добавление двух новых вершин v_1 и v_2 с равновероятными переходами $P(H) = 0.5$ и вероятностью ошибки $E(v_1) = 0.4$ и $E(v_2) = 0$, то итоговая ошибка $E(v)$ никак не изменится $E(v) = 0.5 * 0.4 + 0.5 * 0 = 0.2$, хотя в результате такого разбиения одна из веток дает сто процентный результат классификации.

1.3 Построение сверху-вниз

Функцию $E(T)$ можно использовать для оценки качества уже построенного дерева, а не для выбора одного из множества поддеревьев на этапе разбиения. Таким образом, выращивая дерево, можно говорить о решении задачи минимизации функции $E(T_i)$, где T_i — множество всех возможных

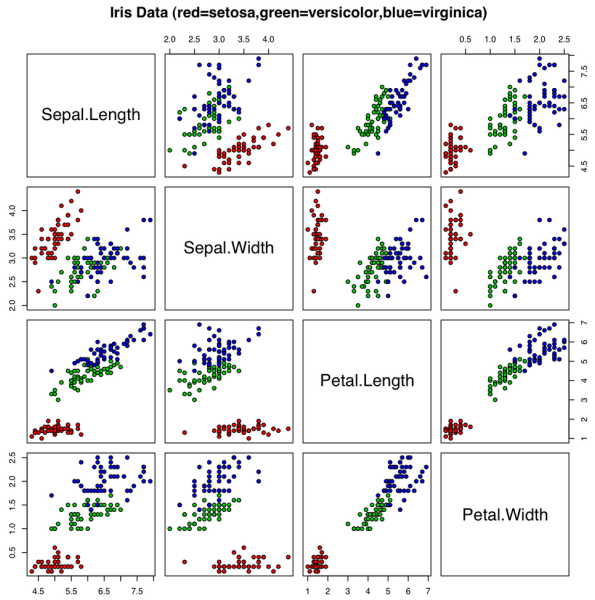
классификационных деревьев.

Пусть для некоторого предиката $h \in H$, терм $T(l, h)$ определяет дерево, в котором у листа l существует два новых дочерних элемента, переход в которые осуществляется по h. Тогда алгоритм построения можно определить следующим образом: на вход подается множество H и максимальный размер дерева t, построение начинается с дерева T_0, состоящим из одной вершины, и далее по алгоритму

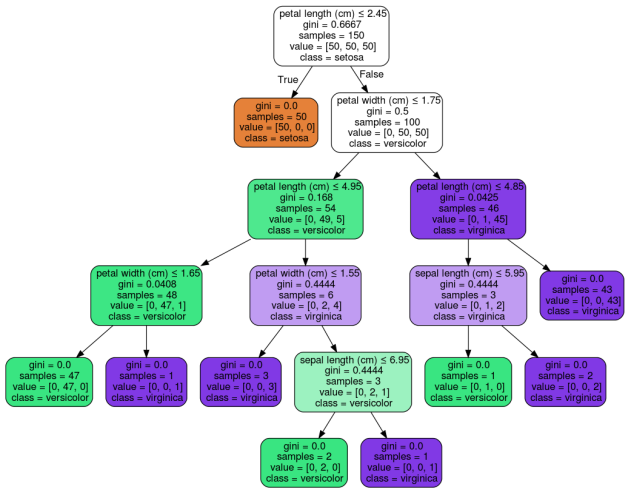
1. Закончить выращивание, если размер дерева больше установленного ограничения t

2. Построить новое дерево $T_{i+1} \leftarrow min_{l \in T, h \in H}(E(T_i) - E(T_i(l, h)))$

По такому принципу работают алгоритмы CART и C4.5, названный в 2008 году алгоритмом машинного обучения №1 [3]. Эти алгоритмы доступны в качестве готовых библиотек для языка программирования Python. С их помощью можно легко вырастить дерево для задачи классификации ирисов Фишера (рисунок 3) — набора данных, на примере которого Рональд Фишер в 1936 году продемонстрировал работу разработанного им метода дискриминантного анализа [4].



а)



б)

Рисунок 3 – а) Диаграмма рассеяния ирисов Фишера
б) Дерево принятия решений для классификации ирисов Фишера

1.4 Процедура обучения

При использовании автоматических методов построения деревьев принятия решений отдельное внимание необходимо уделять обучающей выборке чтобы не допустить переобучения (overfitting) — ситуации, когда во входном наборе данных обнаруживаются некоторые случайные закономерности, отсутствующие в генеральной совокупности. В противном случае, полученная модель является недостаточно общей.

Один из подходов борьбы с проблемой переобучения заключается в заготовке качественных наборов данных, разделенных на три группы:

- Обучающее множество, которое используется в алгоритме рекурсивного секционирования.
- Аттестационное множество, для усовершенствования дерева решений, полученного в результате обучения.
- Проверочное множество, позволяющее выполнить окончательную проверку результатов обучения для подтверждения применимости полученного дерева решений.

1.5 Отсечение веток

Отсечение частей дерева решений (pruning) является алгоритмом улучшения качества и осуществляется на этапе обработки, который следует за обучением. Для его работы необходимо иметь аттестационное множество, отличное от обучающей выборки. Метод основан на предположении, что в дереве существуют ребра, убрав которые произойдет увеличение точности.

Для того чтобы осуществить подобную проверку, необходимо знать, какое промежуточное значение накапливается в каждом узле принятия решений. Далее происходит тестирование дерева на всем аттестационном множестве с подсчетом успешно классифицированных наборов данных в каждой вершине. Если окажется, что некоторый узел имеет лучшую оценку, чем сумма всех его сыновей, то происходит усечение дерева (рисунок 4).

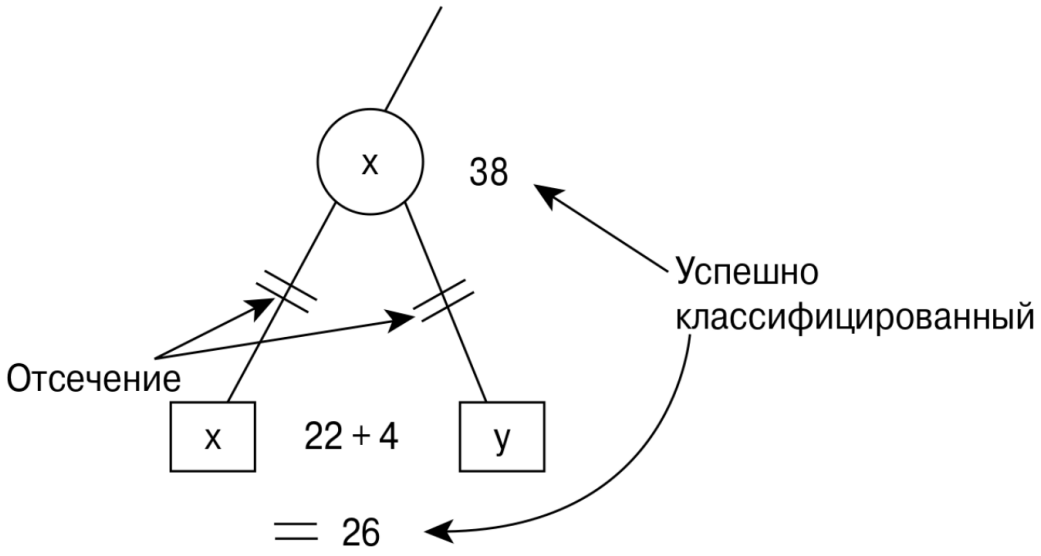


Рисунок 4 – Отсечение ребер дерева после подсчета успешно классифицированных примеров

1.6 Случайный лес

Метод случайного леса (random forest) основан на использовании комитета (ансамбля) деревьев принятия решений. Он был предложен Leo Breiman [5], одним из создателей алгоритма для построения бинарного дерева решений (CART). Метод дает высокое качество результата, лучше чем у нейронных сетей [6], эффективно обрабатывает данные с большим числом признаков классов, обладает возможностью оценивать значимости отдельных признаков в модели.

При обучении создается выборка из N примеров, а также задаются размерность пространства признаков M и дополнительный параметр m (для задач классификации обычно выбирается $m \approx sqrt(M)$). Все деревья комитета строятся независимо друг от друга по следующей процедуре:

- Генерируется случайное подмножество с повторениями размером N из множества обучающей выборки.

- Выращивается решающее дерево, классифицирующее примеры данного подмножества, причём в ходе создания очередного узла выбирается признак не из всего пространства признаков размерности M, а лишь из m случайно выбранных.

- Дерево строится до полного исчерпания подмножества примеров и не обрабатывается процедурами улучшения результата (такими как отсечение веток).

Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает тот, за который проголосовало наибольшее число деревьев. Размерность леса подбирается таким образом, чтобы минимизировать ошибку классификатора на тестовой выборке.

Случайные леса могут быть естественным образом использованы для оценки важности переменных в задачах регрессии и классификации. Меняя параметры выборки, и определяя точки максимумов можно установить какие из них являются более важными для тренировочного набора.

Использование случайного леса для задачи классификации изображений вместе с алгоритмами выделения регионов интереса (ROI — Region Of Interest) дает хорошие результаты [7].

1.7 Метод Виолы-Джонса

Рассмотрим пример использования деревьев принятия решений для задачи распознавания лиц на фотографии в рамках алгоритма Виолы—Джонса, позволяющего обнаруживать объекты на изображениях. Идея метода заключается в сканировании исходной картинки окном поиска, применяя классификатор к каждому полученному положению.

При работе с данными используется интегральное представление фотографии, которое позволяет быстро рассчитывать суммарную яркость произвольного прямоугольника на заданной картинке. Для этого заполняется матрица такой же размерности как и исходное изображение, в каждой ячейке которой хранится сумма интенсивностей всех пикселей, находящихся левее и выше данного элемента. Рассчет таблицы производится по формуле 1:

$$L(x, y) = \sum_{i=0,j=0}^{i \leq x, j \leq y} I(i, j) \tag{1}$$

где $I(i, j)$ — яркость пиксела с координатами i, j. Очевидно, что зная значения соседних ячеек таблицы L, рассчет текущей позиции x, y можно выпол-

нить за константное время (формула 2) и вычислить интегральное изображение за один проход и линейное, относительно размера фотографии, время.

$$L(x,y) = I(x,y) - L(x-1,y-1) + L(x,y-1) + L(x-1,y) \qquad (2)$$

Определим признак как функцию f, переводящую объект $x \in X$ в элемент d из множества допустимых значений признака D_f. Пусть имеется некоторый набор признаков $f_1, f_2, \ldots, f_n$, введем также понятие вектора признаков для объекта $x : (f_1(x), f_2(x), \ldots, f_n(x))$, который будем называть признаковым описанием $x \in X$. В зависимости от значений множества D_f, признаки делятся на:

- бинарные признаки, $D_f = \{0, 1\}$
- номинальные признаки, D_f — конечное множество
- порядковые признаки, D_f — конечное упорядоченное множество
- количественные признаки, D_f — множество действительных чисел

В стандартном методе Виолы-Джонса используются количественные признаки, вычисляемые с использованием примитивов Хаара, представляющих собой прямоугольники, разделенные на черные и белые части (рисунок 5).

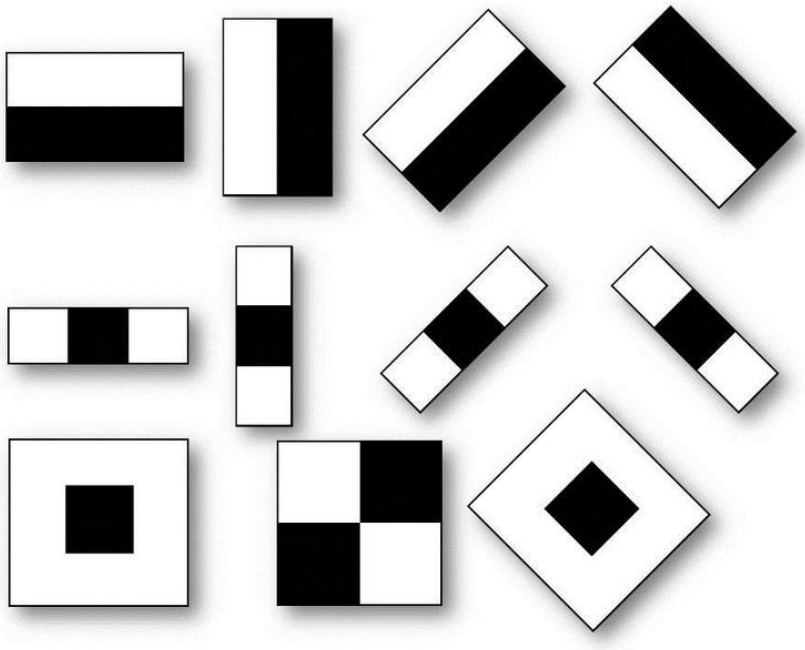


Рисунок 5 – Примитивы Хаара

Значение признака вычисляется по формуле 3, где X — точки светлой части признака, а Y — темной.

$$F = \sum_{i,j \in X} L(i,j) - \sum_{i,j \in Y} L(i,j) \tag{3}$$

Таким образом, идея алгоритма заключается в постепенном проходе всего изображения заданным окном поиска, которое в каждом своем положении передается на вход дереву принятия решений, выносящему вердикт о том что обнаружено на участке фотографии. Очевидно что не всегда в окно поиска попадет объект который необходимо классифицировать, поэтому, при выращивании дерева добавляется специальный выходной класс, отвечающий за «пустоту» (на указанном участке ничего не найдено).

2 Нейронные сети

Под нейронными сетями понимают математическую модель самообучающейся системы, имитирующей деятельность нервных клеток в биологических нейронных сетях. Понятие возникло при изучении процессов, протекающих в мозге, и при попытке их моделирования. Несмотря на большое разнообразие вариантов нейронных сетей, все они состоят из большого числа связанных между собой однотипных элементов — нейронов, иммитирующих одноименные клетки живых организмов.

Искусственный нейрон, так же, как и живой, состоит из синапсов, связывающих его входы с ядром, которое осуществляет обработку входных сигналов. Связь с нейронами следующего слоя происходит через аксон. Каждый синапс имеет вес, определяющий насколько соответствующий вход нейрона влияет на его состояние, вычисляемое по формуле 4,

$$S = \sum_{i=1}^{n} x_i w_i \tag{4}$$

где n — число входов нейрона, x_i — значение i-го входа нейрона, w_i — вес i-го синапса. Значение аксона вычисляется как $f(S)$. Функция f называется активационной и может быть своей у каждого нейрона сети. На рисунке 6 показан пример нейросети, состоящей из трех слоев: входные нейроны (зеленый цвет), скрытые (синий) и выходной нейрон (желтый).

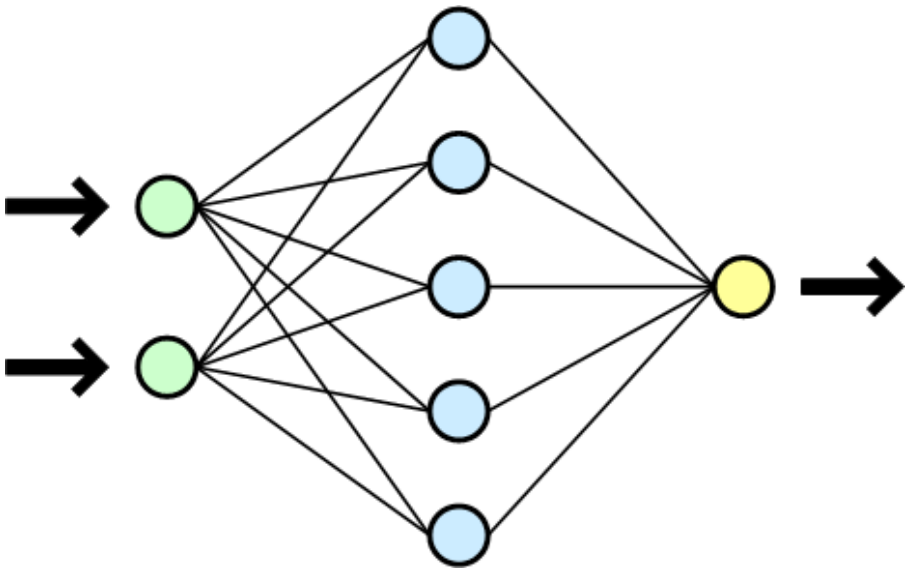


Рисунок 6 – Схема простой нейросети

Нейронные сети активно используются в задачах распознавания образов и предсказывания. Их часто применяют в области машинного обучения, называющейся глубокое обучение (deep learning), занимающейся моделированием высокоуровневых абстракций данных, используя системы со сложной архитектурой из множества нелинейных трансформаций. Под глубиной понимается максимальная длина между входным и выходным узлами графа вычислений модели.

2.1 Нейронные сети обратного распространения

Алгоритм обратного распространения ошибки (back propagation) является одним из методов обучения многослойных нейронных сетей прямого распространения, называемых также многослойными персептронами. Данный подход предполагает два прохода по всем слоям сети: прямой, при котором вычисляется реакция на заданный входной образ, и обратный, редактирующий синаптические веса с целью максимального приближения выходного сигнала сети к желаемому.

Рассмотрим работу алгоритма на примере обучения нейронной сети, представленной на рисунке 7.

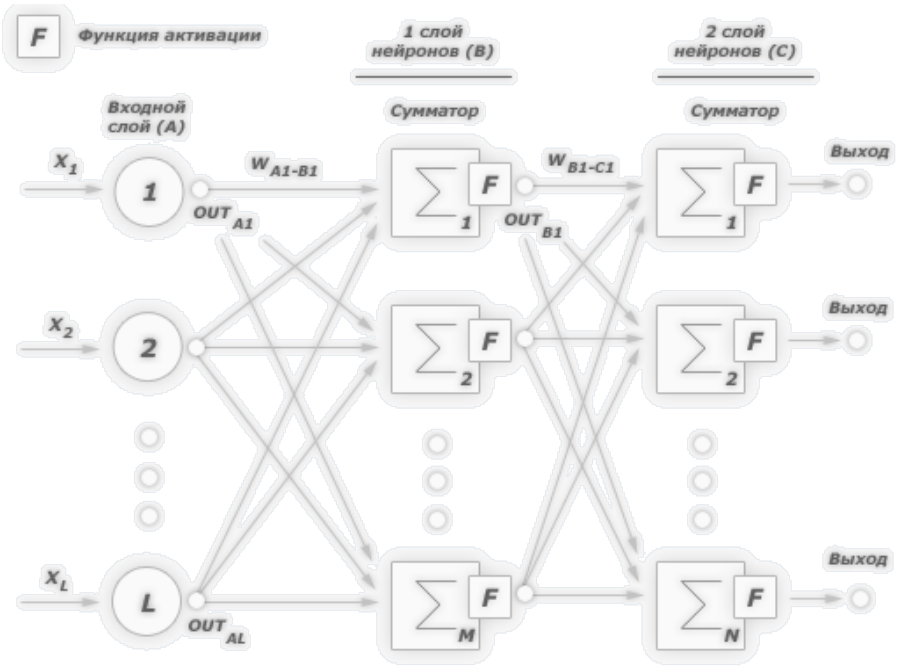


Рисунок 7 – Пример двухслойного персептрона

В качестве активационной, используется сигмоидальная логистическая функция (5), где α — параметром наклона, изменяя который можно управлять крутизной графика. Сигмоид нормирует входной сигнал, возвращая значение из диапазона от нуля до единицы. Для алгоритма обратного распространения ошибки от активационной функции требуется только выполнения условия всюду дифференцируемости. Сигмоид удовлетворяет этому требованию, а его дополнительное преимущество состоит в автоматическом контроле усиления: для слабых сигналов кривая вход-выход имеет сильный наклон, дающий большое усиление, но когда величина сигнала становится больше, усиление падает. Таким образом, большие сигналы воспринимаются сетью без насыщения, а слабые проходят без чрезмерного ослабления.

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \tag{5}$$

Определим понятие обучающей пары — это два вектора, один из которых задает входной сигнал для нейронной сети, а второй отвечает за требуемый выход. Обучение происходит на множестве обучающих пар, его целью является такая настройка синаптических весов, чтобы суммарное отклонение (функция ошибки) от желаемого результата было минимально. Ошибку нейронной сети для каждого теста обучающей выборки можно вычислить по методу наименьших квадратов (6),

$$E(w) = \frac{1}{2} \sum_{i=1}^{p} (d_i - y_i)^2 \tag{6}$$

где y_i — значение i-го выхода нейросети, d_i — ожидаемое значение, p — число нейронов в выходном слое.

Таким образом, ставится задача многомерной минимизации, для решения которой используется метод градиентного спуска. На основании каждой обучающей пары происходит корректировка синаптических весов сети w_{ij} на некоторое число (7),

$$\Delta w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}} \tag{7}$$

где $0 < \eta < 1$ — множитель, задающий скорость движения вдоль направления аниградиента.

В рассматриваемом примере веса $w_{B_i-C_j}$ влияют на выход сети только как часть суммы $S_{C_j} = \sum w_{B_i-C_j} x_{C_j}$ по входам j-го узла, поэтому

$$\frac{\delta E}{\delta w_{B_i-C_j}} = \frac{\delta E}{\delta S_{C_j}} \frac{\delta S_{C_j}}{\delta w_{B_i-C_j}} = x_{C_j} \frac{\delta E}{\delta S_{C_j}}$$

Так как сейчас мы рассматриваем выходной слой сети C, следовательно, S_{C_j} влияет на общую ошибку только в рамках выхода своего узла C_j:

$$\frac{\delta E}{\delta S_{C_j}} = \frac{\delta E}{\delta y_j} \frac{\delta y_j}{\delta S_{C_j}} = \left(\frac{\delta}{\delta y_j} \frac{1}{2} \sum_{i \in C} (d_i - y_i)^2 \right) \left(\frac{\delta f(S_{C_j})}{\delta S_{C_j}} \right) =$$
$$\left(\frac{1}{2} \frac{\delta}{\delta y_j} (d_j - y_j)^2 \right) (y_j(1 - y_j))\, \alpha = -\alpha y_j (1 - y_j)(d_j - y_j) \tag{8}$$

Введем обозначение $Children(X_j)$ определяющее множество выходов для j-го нейрона слоя X. Тогда для внутренних слоев сети, для конкретности рассмотрим A, получается:

$$\frac{\delta E}{\delta S_{A_j}} = \sum_{k \in Children(A_j)} \frac{\delta E}{\delta S_k} \frac{\delta S_k}{\delta S_{A_j}}$$

$$\frac{\delta S_k}{\delta S_{A_j}} = \frac{\delta S_k}{\delta y_{A_j}} \frac{\delta y_{A_j}}{\delta S_{A_j}} = w_{A_j-B_k} \frac{\delta y_{A_j}}{\delta S_{A_j}} = \alpha w_{A_j-B_k} y_{A_j} (1 - y_{A_j}) \tag{9}$$

Таким образом, корректировка синаптических весов для выходного слоя нейронов происходит по формуле (8), а для внутренних (9). Необходимо отметить, что простейший метод градиентного спуска, рассмотренный выше, неэффективен если производные по различным весам сильно отличаются. В этом случае для плавного уменьшения ошибки надо выбирать очень маленькую скорость обучения, но при этом существенно возрастает время работы алгоритма. Для выхода из этой ситуации можно применить метод введения момента, когда со временем происходит изменение влияния градиента на корректировку весов. Такой подход также используется для выхода из точек локального минимума.

2.2 Рекуррентные нейронные сети

Рекуррентными нейросетями (RNN — Recurrent Neural Network) называются такие сети, в которых выходы нейронов последующих слоев имеют синаптические соединения с нейронами расположенными в предшествующих слоях. Наличие обратных связей позволяет запоминать и воспроизводить целые последовательности реакций на один стимул. С точки зрения программирования в таких сетях появляется аналог циклического выполнения, а с точки зрения систем такая сеть эквивалентна конечному автомату.

Существуют различные варианты архитектур рекуррентных нейросетей, например сеть Джордана, в которой нейроны выходного слоя соединены посредством специальных входных нейронов (contextunits — контекстные нейроны) с нейронами промежуточного слоя (рисунок 8).

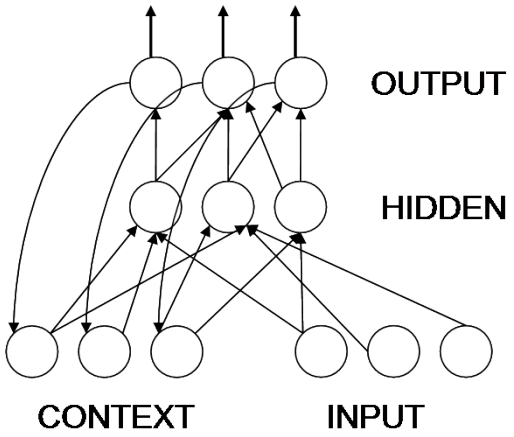


Рисунок 8 – Архитектура рекуррентной сети Джордана

Она была одной из первых рекуррентных нейронных сетей и ее главной идеей было дать сети видеть свой выходной образ на предыдущем шаге. Идеи, заложенные в этой модели были позже усовершенствованы в сеть Элмана, которая так же как и сеть Джордана получается из многослойного перцептрона введением обратных связей, с тем лишь исключением, что связи идут не от выхода сети, а от выходов внутренних нейронов, что позволяет учесть предысторию наблюдаемых процессов и накапливать информацию для выработки правильной стратегии управления. Главной особенностью такой сети является запоминание последовательностей.

Главное преимущество сетей Элмана состоит в том, что число контекстных нейронов определяется не размерностью выхода (как в сети Джордана), а количеством скрытых нейронов, что делает ее более гибкой, так как мож-

но легко добавить или убрать скрытые нейроны, в отличие от количества выходов.

2.3 Задача распознавания изображений

Рассмотрим задачу распознавания рукописных цифр. Необходимо создать и обучить нейросеть, которая по заданному изображению активирует один из дести выходов. В качестве входных данных рассматривается база рукописных цифр MNIST [8], содержащая шестьдесят тысяч обучающих пар (изображение — метка) и десяти тысяч тестовых (изображения без меток). Картинки нормализованы по размеру и отцентрованы. Размер каждой цифры не более 20×20 пикселей, и все они вписаны в квадрат размером 28×28.

При использовании нейронных сетей для решения этой задачи возникает ряд вопросов, связанных с тем как подать данные в сеть и какую архитектуру (количество скрытых слоев, уровень связности) использовать. На текущий момент до сих пор не существует методов, позволяющих однозначно определить структуру и состав нейросети исходя из описания задачи. Есть множество различных методик редукции (например OBD [9]), а также разные эвристики и эмпирические правила, одно из которых гласит, что количество нейронов в скрытом слое должно быть хотя бы на порядок больше количества входов.

Рассмотрим подход загрузки изображения в нейросеть заключающийся в представлении двумерной матрицы исходной картинки в виде одномерного вектора. Согласно описанной выше эвристике и использовании полносвязной нейронной сети получается что итоговая модель будет иметь $\prod_{i=1}^{N} len(X_i) \times len(X_{i+1})$ связей, где N — количество слоев, а $len(X)$ — размер слоя X. Таким образом, для рассматриваемой задачи распознавания рукописной цифры получается $28 \times 28 = 784$ нейрона на входном слое, а при использовании двух слоев по десять и пять тысяч нейронов в каждом, около шестидесяти миллионов связей (десять миллионов между входным и первым скрытым слоем, пятьдесят миллионов между первым и вторым и пятьдесят тысяч между вторым и выходным).

При преобразовании изображения в линейную цепочку байт, происходит потеря взаимосвязи между отдельными частями входной картинки, а решае-

мая задача распознавания подразумевает умение нейросети быть устойчивой к небольшим сдвигам, поворотам и изменению масштаба изображения, а значит она должна извлекать из данных некие инварианты, не зависящие от почерка того или иного человека. Таким образом, полносвязные нейронные сети обратного распространения не очень хорошо подходят для распознавания образов, так как являются очень вычислительно сложным решением и, в тоже время, не способны корректно справляться с различными искажениями входных изображений.

2.4 Сверточные нейронные сети

Сверточная нейронная сеть (CNN — Convolutional Neural Network) — это специальная архитектура искусственных нейронных сетей, нацеленная на эффективное распознавание изображений и предложенная Яном Лекуном [10], вдохновленным работами нобелевских лауреатов в области медицины Torsten Nils Wiesel и David H. Hubel. Эти ученые исследовали зрительную кору головного мозга кошки и обнаружили, что существуют так называемые простые клетки, которые особо сильно реагируют на прямые линии под разными углами и сложные клетки, которые реагирую на движение линий в одном направлении.

Идея архитектуры сети заключается в чередовании сверточных слоев (C-layers), субдискретизирующих слоев (S-layers) и наличии полносвязных (F-layers) слоев на выходе, структура — однонаправленная (без обратных связей). В качестве активационной используется любая функция по выбору исследователя. Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

При данном подходе рассматриваются три основные парадигмы:

- Локальное восприятие
- Разделяемые веса
- Субдискретизация

Локальное восприятие подразумевает, что на вход одного нейрона подается не все изображение (или выходы предыдущего слоя), а лишь некоторая его область. Такой подход позволил избежать потерю топологии рассматриваемой картинки, будет сохранена взаимосвязь между ее отдельными частями.

Концепция разделяемых весов предполагает, что для большого количества связей используется очень небольшой набор весов. Каждый из нейронов следующего слоя принимает на вход не все, а только некоторый участок обрабатываемого изображения. Такие узлы называют ядрами (kernels), они составляют сверточный слой нейросети, выполняют операцию свертки и порождают карту признаков (рисунок 9), элементы которой обозначают степень похожести фрагмента изображения на закодированный ядром фильтр. Такое искусственно введенное ограничение на веса улучшает обобщающие свойства сети (generalization), что в итоге позитивно сказывается на ее способности находить инварианты в изображении и реагировать главным образом на них, не обращая внимания на прочий шум.

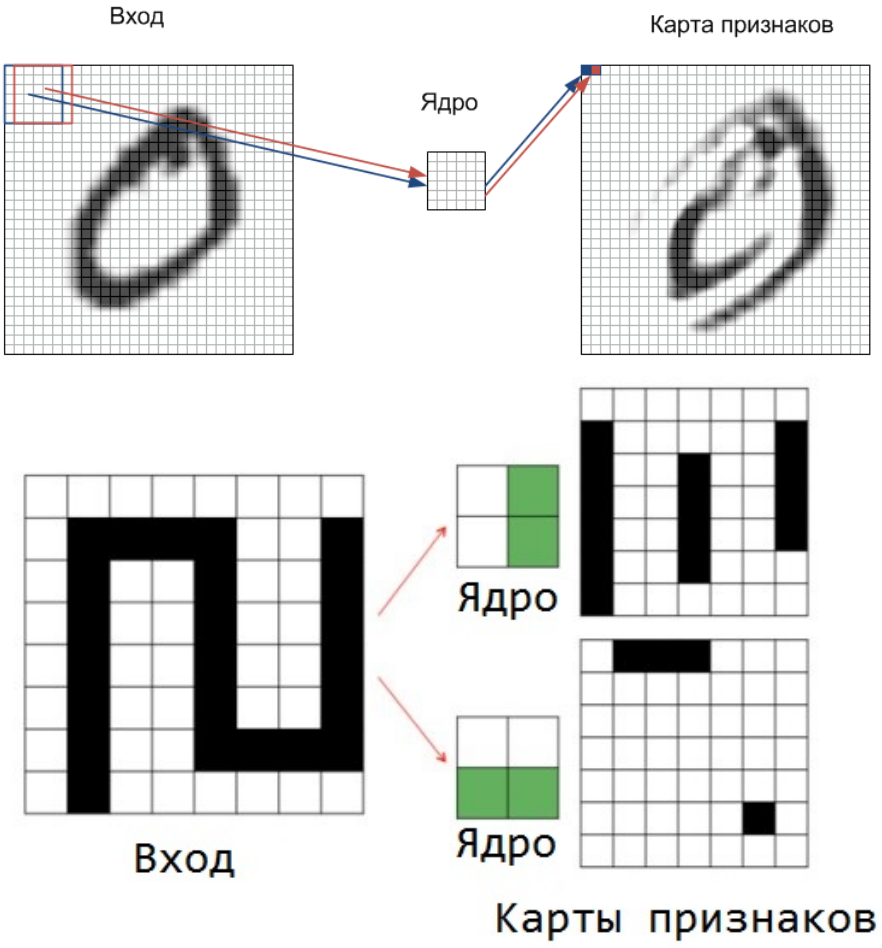


Рисунок 9 – Примеры распространения сигнала в С-слое

Суть субдискретизации и S-слоев заключается в уменьшении пространственной размерности изображения для обеспечения инвариантности к масштабу. Чередование слоев позволяет составлять карты признаков из предыдущих слоев, что на практике означает способность распознавания сложных иерархий признаков.

По мере продвижения вглубь сверточной нейронной сети, возникает вопрос уточнения информации на картах признаков, так как если признак присутствует на предыдущем слое, полученная в результате карта содержит ключевые пикселы, окруженные нечетким «ореолом», в рамках которого признак может быть определен не до конца. Для решения этой проблемы используют метод, называемый max-объединением (max pooling), который состоит в разделении карты признаков на непересекающиеся участки и выделении на них нейронов с максимальной активностью (рисунок 10). Max-объединение карты признаков делает процесс распознавания более точным, избавляясь от ненужных разводов и сокращая число параметров сверточной нейросети.

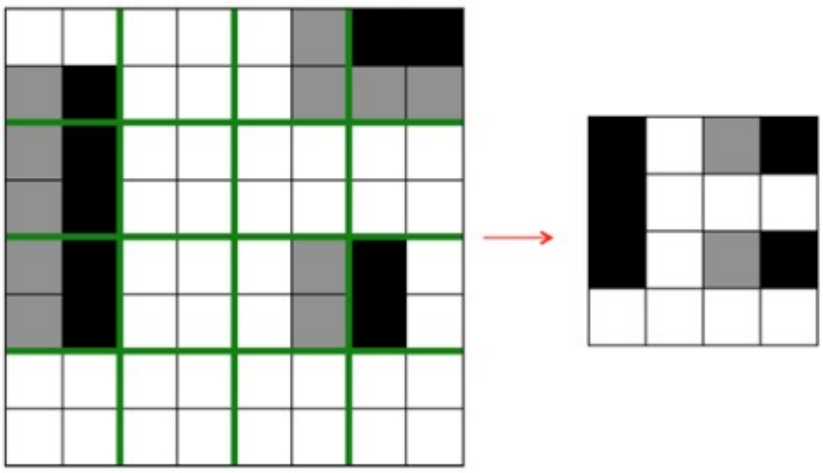


Рисунок 10 – Иллюстрация работы метода max pooling выделения нейронов с максимальной активностью

2.4.1 Обучение

Аналогично пункту 2.1 рассмотрим процесс обучения сверточных нейронных сетей. Точно также будем использовать среднеквадратичную функцию ошибки (6) для очередного теста. Ошибку на всей обучающей выборки определим как среднее арифметическое по ошибкам для всех обучающих пар.

Такую усредненную ошибку обозначим как E, а за E_p примем ошибку на p-ой обучающей паре.

Минимизация функции E_p, как и в методе обратного распространения ошибки, происходит при помощи градиентного спуска. Рассмотрим пример сети с всего одним набором весов, разложим в ряд Тейлора функцию ошибки E_p (10), где W_c — некоторое начальное значение веса.

$$E(W) = E(W_c) + (W - W_c)\frac{dE(W_c)}{dW} + \frac{1}{2}(w - W_c)^2\frac{d^2E(W_c)}{dW^2} + \dots \qquad (10)$$

Теперь возьмем производную функции ошибки по весам и отбросим члены выше 2-го порядка (11):

$$\frac{dE(W)}{dW} = \frac{dE(W_c)}{dW} + (W - W_c)\frac{d^2E(W_c)}{dW^2} \qquad (11)$$

Таким образом, вес, при котором значение функции ошибки будет минимальным можно вычислить следующим образом (12):

$$W^* = W_c - \left(\frac{d^2E(W_c)}{dW^2}\right)^{-1}\frac{dE(W_c)}{dW} \qquad (12)$$

То есть, оптимальный вес вычисляется как текущий минус производная ошибки по весу, деленная на вторую производную функции ошибки. Для многомерного случая используется тот же самый алгоритм, только первая производная превращается в градиент, а вторая соответственно в Гессиан. И здесь возможно два варианта: если опустить вторую производную, то получится алгоритм наискорейшего градиентного спуска, а иначе — обычный, требующий очень больших затрат на вычисление Гессиана, поэтому его заменяют чем-то более простым, например, один из самых известных и успешных методов — алгоритм Левенберга-Марквардта для использования вместо Гессиана его аппроксимации с помощью квадрадного Якобиана. Но, необходимо отметить, что алгоритм Левенберга-Марквардта требует обработки всей обучающей выборки, тогда как метод наискорейшего градиентного спуска может работать с каждой отдельно взятой обучающей парой.

2.4.2 Дообучение

Рассматриваемая архитектура нейронной сети нацелена на автоматическое выделение опорных признаков для классификации объектов из заданной предметной области. Благодаря чередованию сверточных и субдискретизирующих слоев эту задачу получается успешно решать, однако процесс обучения требует огромных вычислительных ресурсов. Возникает вопрос, возможно ли при изменении (расширении или наоборот уменьшении) предметной области не производить полное переобучение нейронной сети и при этом не допустить сильной потери точности.

Методы для решения данной проблемы используют подход частичной корректировкой весов (fine tuning), а конкретно — вносят существенные изменения лишь в последние полносвязные слои. Такие идеи основаны на сохранении выработанных сетью признаков, перестраивая лишь реакцию на них.

В работе [11] рассматривается пример перестроения весов, обученной на широкой выборке нейронной сети, для работы с узкой выборкой. Важно отметить, что рассматриваемая предметная область не менялась. У обученной на классификацию тысячи различных образов сети меняются веса выходного слоя на случайные числа, после чего происходит корректировка методом обратного распространения ошибки на новой обучающей выборке со скоростью обучения верхних слоев на два порядка выше, чем у последних. Результаты эксперимента представлены на рисунке 11.

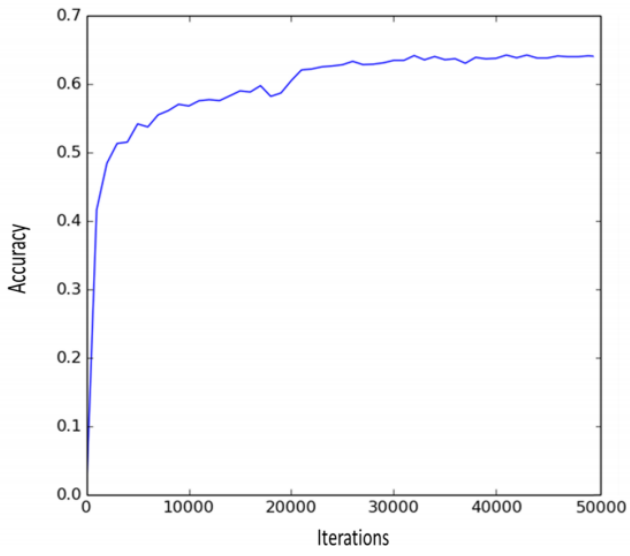


Рисунок 11 – Изменение точности при переобучении нейронной сети

Хорошо видно, как быстро растет точность уже на первых итерациях корректировки весов, а после сорока тысяч происходит выход на плато. Таким образом, было показано что однажды обученную нейросеть можно повторно использовать на немного измененной предметной области без полного переобучения.

Используемые обозначения

ANN — Artificial Neural Network, искуственная нейронная сеть (тоже самое что и NN)
Back propagation — алгоритм обратного распространения ошибки обучения многослойных нейронных сетей
CART — Classification And Regression Trees, алгоритм для построения бинарного дерева решений
CNN — Convolutional Neural Network, сверточная нейронная сеть
Data mining — автоматические методы интеллектуального анализа
Decision tree — деревья принятия решений
Deep learning — область машинного обучения, занимающаяся моделированием высокоуровневых абстракций данных, используя системы со сложной архитектурой из множества нелинейных трансформаций
Fine tuning — алгоритмы частичной корректировки коэффициентов модели при ее дообучении
Generalization — обобщающие свойства CNN
Information extraction — методы автоматического выделения информации
Kernels — в контексте CNN, узлы сверточного слоя
Logistic function — сигмоидальная функция
Logistic neuron — нейрон с сигмоидальной активационной функцией
ML — Machine Learning, машинное обучение
NN — Neural Network, нейронная сеть
MNIST — база данных рукописных цифр
Pruning — отсечение веток дерева решений, алгоритм улучшения качества
OBD — Optimal Brain Damage, метод редукции нейросети
OCR — Optical Character Recognition, задача распознавания рукописного текста
Overfitting — переобучение
Random forest — комитета решающих деревьев для задач классификации и регрессии
RNN — Recurrent Neural Network, вид нейронных сетей, в которых имеется обратная связь
ROI — Region Of Interest

Список литературы

[1] Learning hierarchical features for scene labeling / C. Farabet, C. Couprie, L. Najman, Y. LeCun // *Pattern Analysis and Machine Intelligence, IEEE Transactions on.* — 2013. — Vol. 35, no. 8. — Pp. 1915–1929.

[2] *Quinlan, J. R.* Decision trees and decision-making / J. R. Quinlan // *Systems, Man and Cybernetics, IEEE Transactions on.* — 1990. — Vol. 20, no. 2. — Pp. 339–346.

[3] Top 10 algorithms in data mining / X. Wu, V. Kumar, J. R. Quinlan et al. // *Knowledge and Information Systems.* — 2008. — Vol. 14, no. 1. — Pp. 1–37.

[4] *Fisher, R. A.* The use of multiple measurements in taxonomic problems / R. A. Fisher // *Annals of eugenics.* — 1936. — Vol. 7, no. 2. — Pp. 179–188.

[5] *Breiman, L.* Random forests / L. Breiman // *Machine learning.* — 2001. — Vol. 45, no. 1. — Pp. 5–32.

[6] *Niculescu-Mizil, A.* An empirical comparison of supervised learning algorithms using different performance metrics: Tech. rep. / A. Niculescu-Mizil, R. Caruana: Cornell University, 2005.

[7] *Bosch, A.* Image classification using random forests and ferns / A. Bosch, A. Zisserman, X. Munoz // Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on / IEEE. — 2007. — Pp. 1–8.

[8] *LeCun, Y.* The mnist database of handwritten digits. — 1998.

[9] *Chauvin, Y.* Advances in neural information processing systems 2 / Y. Chauvin, D. Touretzky // *chapter Dynamic behavior of constrained back propagation networks.* — 1989. — Pp. 642–649.

[10] Comparison of learning algorithms for handwritten digit recognition / Y. LeCun, L. Jackel, L. Bottou et al.

[11] *Reyes, A. K.* Fine-tuning deep convolutional networks for plant recognition / A. K. Reyes, J. C. Caicedo, J. E. Camargo // Working notes of CLEF 2015 conference. — 2015.