Санкт-Петербургский государственный университет

Математико-механический факультет

Прикладная математика и информатика

Ковальчуков Александр Алексеевич

1 курс, 123 группа

Курсовая работа

Обзор методов машинного обучения с учителем

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

Санкт-Петербург
2019

Введение

> Машинное обучение - область исследований, которая наделяет компьютеры возможностью проявить поведение, которое не было заложено в них явно.
>
> Артур Сэмюэл, 1959 год[1]

Эта работа написана с целью дать краткий обзор машинного обучения - одной из дисциплин искусственного интеллекта, возникшей в середине XX века на стыке информатики и статистики. Машинное обучение предоставляет медоды приближённого решения задач, плохо или совсем не поддающиеся решению аналитическим путём. В основе этих методов лежат математические модели, которые обучаясь на наборах данных генерируют ассоциативные правила, позволяющие дать ответ на задачу в общем случае. Благодаря бурному развитию цифровой электроники, методы вышли за пределы лабораторий и получили широкое распространение в повседневной жизни. Так, персонализированная реклама, автокоррекция текста, машинный перевод и многие другие приложения в той или иной степени задействуют ML. Всплеск количества публикаций по теме машинного обучения приходится на начало 2000. Растущий интерес к области можно соотнести с развитием интернета и накоплением в нём разнообразных данных.

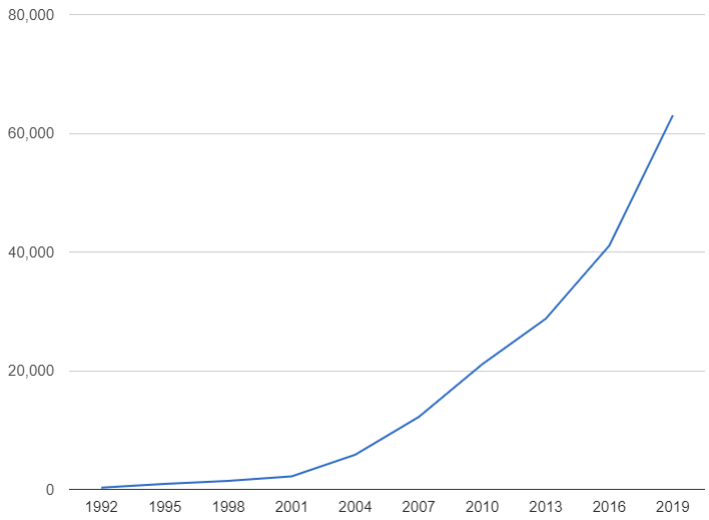

Рис. 1: График количества публикаций с 1992 по 2019 год.
Построен по количеству побликаций в электронной библиотеке IEEE explore
Запрос производился по ключевым словам ”machine learning”

Условные обозначения

- **ML**: Machine Learning
- **AI**: Artificial Intelligence
- **SVM**: Support Vector Machine
- **ANN**: Artificial Neural Network
- **CNN**: Convolutional Neural Network
- **CPU**: Central Processor Unit
- **GPU**: Graphical Processor Unit

Мотивация

Основная область применения машинного обучения - это приближённое решение задач, и среди них принято выделять несколько основных:

- **Задача классификации** - отнесение объекта выборки к одному из нескольких классов по ассоциации с уже известными примерами.
- **Кластеризация** - выработка правила, упорядочевающего объекты исходной выборки в сравнительно однородные классы. Существенное различие с задачей классификации состоит в том, что модели не предоставляется тренировочных данных или иных подсказок относительно искомой закономерности в данных.
- **Задача регрессии** - поиск зависимости между значением целовой переменной и значениями одной или нескольких независимых величин.
- **Снижение размерности** - преобразование данных, состоящее в уменьшении числа признаков объекта путём получения главных переменных.
- **Ранжирование** - упорядочивание объектов по некоторому критерию. Нередко сводится к задачам классификации и регресии.
- **Фильтрация выбросов** - повышение качества исходной выборки для дальнейшего применения на чувствительных к выбросам моделей.

Процесс машинного обучения

В общем контексте обучения с учителем процесс ML можно описать как подбор функции $f : X \to y$, которая адекватным образом соотносит входные данные X значению целевой переменной y.

Распространённый на практике процесс можно условно разделить на несколько стадий:

1. Постановка задачи

2. Добыча данных
3. Подготовка данных
4. Подбор модели
5. Обучение модели
6. Проверка адекватности
7. Анализ результатов

Рассмотрим пункты 3 - 6 подробнее.

Подготовка данных

Не все модели способны воспринимать данные в естественной форме, поэтому зачастую требуется серьезная предобработка данных и удаление выбросов - качество модели напрямую зависит от полноты и чистоты тренировочной выборки. В задачи ML инженера входит очистка данных, их преобразование и нормализация.

Подбор модели

Иногда выбор модели определяется поставленными задачами и форматом данных, однако порой оптимальных критериев подбора модели нет. В таких случаях приходится перебрать несколько моделей и выбрать из них лучшую, либо объеденить лучшие в ансамбль.

Обучение модели

На этом этапе происходит настройка внутренних параметров модели с целью поиска закономерности в тренировочных данных. Все имеющие в распоряжении данные необходимо разбить на два непересекающихся набора: на одном модель будет обучаться в соответствии с её алгоритмом обучения, а на втором тестироваться. Особых рекомендаций по соотношению частей разбиения нет, хотя обычно модель обучают на 70% данных, а тестируют на оставшихся 30%. Но при этом чрезвычайно важно, чтобы данные в разбиении были представлены равномерно, чтобы обучение было объективным. В простейшем случае разбиение производится случайным образом.

Проверка адекватности

На этом этапе модели на вход подаются данные, которые не были задействованы в процессе обучения, и по этим данным вычисляется общая ошибка модели.

Матрица несоответствия

В случае классификации по всей тестовой выборке составляется матрица A, ij элемент которой показывает, сколько элементов класса i были классифицированны как класс j. Сумма чисел на её главной диагонали даёт ответ сколько объектов было классифицированно правильно. По этой матрице разными способами можно извлечь численную оценку эффективности модели.

1. **Точность и полнота**
   В простейшем случае за эффективность модели можно принять отношение правильно классифицированных объектов к размеру выборки:

$$Accuracy = \frac{Tr(A)}{\sum a_{ij}}$$

   Однако такая оценка оказывается необъективной, когда один класс в выборке встречается значительно чаще других, и необъективная модель, которая хорошо работает только на нём, получит хорошую оценку.

2. **Точность и полнота**
   Точность в пределах класса - это доля правильно классифицированных объектов среди всех ответов модели по данному классу:

$$Precision_k = \frac{a_{kk}}{\sum\limits_{i=1}^{n} a_{ik}}$$

   Полнота - это доля правильно классифицированных объектов данного класса среди объектов, действительно к нему относящихся.

$$Recall_k = \frac{a_{kk}}{\sum\limits_{j=1}^{n} a_{kj}}$$

   Точность и полнота всего классификатора вычисляется как среднее арифметическое точности и полноты по всем классам.

3. F_1 **- метрика**
   В редких случаях удаётся одновременно достичь и точности и полноты одновременно. Если предпочтение отдаётся только точности или только полноте, то проблемы нет - можно сравнивать модели по одному критерию. Но если важны оба критерия, то оценочной метрикой можно ввести величину, равную среднему гармоническуму между точностью и полнотой:

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2\frac{Precision \times Recall}{Precision + Recall}$$

   Такая величина стремится к 1, если точность и полнота одновременно стремятся к 1, и стремится к нулю, если хотя бы одна из этих величин стремится к нулю.

Среднеквадратическая ошибка

В случае регрессии индикатором качества модели может служить среднеквадратическая ошибка. Чем она меньше, тем модель точнее.

$$RMSE = \frac{\sum\limits_{i=1}^{n} (y_i - f(x_i))^2}{n}$$

Cross-validation

Кросс-валидация - это техника валидации модели, позволяющая предсказать её работу на независимом наборе данных, при фактическом отсутствии такового. Один её цикл состоит в разбиении выборки на 2 части: на одной модель обучается, а на другой проверяется. Чтобы минимизировать ошибку, разные иттерации кросс-валидации производятся на разных разбиениях, а результаты валидации усредняются по всем циклам. Таким образом, модель можно обучать на всех имеющихся данных.
Существует несколько распространённых типов кросс-валидации:

1. **По k блокам**
   Исходный набор разбивается на k блоков, k-1 используются для тренировки, а 1 для валидации. Процесс повторяется k раз, причём блок валидации каждый раз меняется.

2. **Поэлементная кросс-валидация**
   Представляется собой кросс-валидацию по k блокам, где k равно количеству объектов в выборке.

3. **Валидация случайным сэмплированием**
   На каждой иттерации выборка случайным образом разбивается на тренировочный и тестовый набор, но при этом некоторые объекты могут ни разу не попасть в тестовую выборку, а некоторые напротив - попасть в неё несколько раз.

Методы машинного обучения

Существует огромное количество моделей, заточенных под различные задачи, а в пределах каждой из них существуют десятки различных вариаций. Я постараюсь дать обзор самых популярных из них.

Линейная регрессия [2]

Модель разработана задолго до наступления компьютерной эпохи, впервые появившись в работах Лежандра и Гаусса на рубеже XVIII и XIX веков[3]. Суть состоит в определении линейной зависимости между одной целевой и несколькими независимыми переменными. Пусть анализируемый объект имеет n свойств: $x_1, x_2, \dots x_n$. Тогда значение целевой переменной определяется функцией $y = \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n + \varepsilon$ с нормально распределённой случайной величиной ε.

Пусть $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$ - вектор наблюдений зависимой переменной

$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \dots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix}$ - матрица признаков

$\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}$ - вектор случайных величин

Тогда коэффициенты модели ω являются решением неоднородной системы линейных уравнений: $X\omega + \varepsilon = y$

Нелинейная регрессия

Реальные зависимости далеко не всегда хорошо описываются линейными моделями. В некоторых случаях может помочь нелинейная регрессия.
Приведём некоторые часто используемые регрессионные модели:

1. Полиномиальная одномерная регрессия: $y = \sum\limits_{i=0}^{n} \omega_i x^i$. Следует помнить, что полиномы высоких степеней склонны к переобучению
2. Ряд Фурье: $y = a_0 + \sum\limits_{i=1}^{n} (a_i \sin(ix) + b_i \cos(bx))$. Хорошо приближает периодические функции.
3. Рациональный полином: $y = \frac{\sum\limits_{i=0}^{n} a_i x^i}{\sum\limits_{j=0}^{m} a_j x^j}$

Наивный Байесовский классификатор [4]

Модель получила своё название по теореме лежащей в её основе. При классификации используется предположение о том, что все признаки в классе независимы. И хотя реальные признаки чаще всего коррелируют друг с другом, метод на удивление хорошо работает во многих ситуациях. Вероятностная модель устроена следующим образом:
$P(y \mid X)$, где X - вектор признаков.
Используя теорему Байеса, формулу можно переписать в виде:

$$P(y \mid X) = \frac{P(X \mid y)P(y)}{P(X)}$$

И используя независимость признаков,

$$P(y \mid X) = \frac{P(y) \prod\limits_{i=1}^{n} P(x_i \mid y)}{\prod\limits_{i=1}^{n} P(x_i)}$$

После выполнения модели объект относится к классу с наибольшей вероятностью.
К достоинствам метода можно отнести высокую скорость обучения, небольшой размер обучающей выборки, а к недостаткам - достаточно грубое допущение о независимости переменных,

а так же явление, называемое нулевой частотой: если значение какого-то признака ни разу не встречалось в обучающей выборке, то ему будет присвоена нулевая вероятность.

Метод k-ближайших соседей [5]

В основе метода лежит гипотеза компакности - предположение о том, что объекты одного класса похожи между собой: в пространстве объектов они образуют связную ограниченную область. Чтобы формализовать понятие похожести, на множестве объектов вводится метрика, адекватно оценивающая расстояние между объектами, причём её выбор целиком возлагается на плечи исследователя.
Пусть $\rho(x_1, x_2)$ - метрика, x - классифицируемый объект, а $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ - k ближайших к x объектов.
Тогда классифицируемый объект относится к тому классу, к которому относится большинство из его k соседей.

$$f(x) = arg\max_{y \in Y} \sum_{j=1}^{k} [y(x_{i_j}) = y]$$

.

Модель также применима к задачам регресии. Восстановать значение целовой переменной в точке x из пространства признаков можно взяв среднее арифметическое значений k соседних точек $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$:

$$f(x) = \frac{1}{k} \sum_{j=1}^{k} y(x_{i_j})$$

Метод опорных векторов [6]

SVM, первоначально применялась для задач бинарной классификации объектов, расположенных в арифметическом векторном пространстве. В случае линейной разделимости классов естественным решением будет построение гиперплоскости с наибольшим зазором с объектами обучающей выборки, и отнесение всех объектов по одну сторону гиперплоскости к одному классу, а по другую - к другому. Было аналитически доказано, что максимизация зазора способствует более уверенной классификации[7].
Пусть имеется выборка $(x_1, y_1), \ldots, (x_k, y_k)$ $x_i \in \mathbb{R}^n, y \in \{-1, 1\}$, и мы строим разделяющую гиперплоскость $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$, такую что для всех точек одного класса $\langle w, x_i \rangle + b > 0$, а для другого $\langle w, x_j \rangle + b < 0$.
Заметим, что при умножении уравнения плоскости на константу, уравнение не меняется. Поэтому подберём такие w и b, чтобы для ближайших к плоскости точек выполнялось равенство $\langle w, x_i \rangle + b = y_i$. Эти точки называются опорными.
Так 2 класса оказываются разделены полосой $-1 \leq \langle wx \rangle + b \leq 1$. Пусть x_1, x_2 - 2 точки с разных границ полосы.
Тогда ширина полосы равна проекции вектора $x_1 - x_2$ на прямую с направляющей w. Зазор $d = \frac{\langle x_1 - x_2, w \rangle}{||w||} = \frac{\langle x_1, w \rangle - \langle x_2, w \rangle}{||w||} = \frac{(b+1)-(b-1)}{||w||} = \frac{2}{||w||}$ минимален, когда $||w||$ максимальна.
Таким образом построение опримальной гиперплоскости сводится к задаче квадратичного программирования: Минимизировать $||w||$ при условии $y_i \langle w, x_i \rangle + b \geq 1 \quad \forall i \in 1 \ldots k$.

Но выборка редко бывает линейно разделимой. Ключевой идеей, позволяющей применить SVM в этом случае, является перевод исходных векторов в пространство более высокой размерности, где они будут линейно разделимы. Нужно подобрать отображение $\phi : \mathbb{R}^n \to H$ в пространство достаточно большой размерности, чтобы задача минимизации $||\phi(w)||$ при $y_i\langle\phi(w), \phi(x_i)\rangle + b \geq 1 \quad \forall i \in 1 \dots k$ в нём имела решение. Естественным ограничением на пространство H является наличие в нём скалярного произведения.

Решающие деревья[8]

Благодаря своей простоте и легкой интерпретируемости, решающее дерево является одной из самых распространённый моделей, используемых в машинном обучении. Оно представляет собой связный граф без циклов, в каждом промежуточном узле которого расположено решающее правило, а в листьях - метки классов. Деревья строятся так, чтобы по мере спуска классифицируемого объекта уменьшалась его классовая неопределённость. Для этого в корневой вершине происходит разбиение выборки X на несколько частей $X_1, \dots X_k$, для каждой из которой вычисляется индекс неоднородности $H(X)$. Критерием качества разбиения служит уменьшение неоднородности $\Delta H(X) = H(X) - \sum\limits_{i=1}^{k} H(X_i)\frac{|X_i|}{|X|}$ и чем оно больше, тем лучше разбиение.

На практике часто используются следующие индексы неоднородности, где $K_1, K_2, \dots K_L$ - метки классов:

- Энтропийный индекс неоднородности:
  $H(X) = -\sum\limits_{i=1}^{L} P(K_i)\log_2(P(K_i))$,
  где $P(K_i)$ - доля объектов класса K_i в выборке X. При этом полагается, что $0\log_2 0 = 0$
- Индекс Джинни:
  $H(X) = 1 - \sum\limits_{i=1}^{L} P(K_i)^2$
- Индекс ошибочной классификации:
  $H(X) = 1 - \max\limits_{1\dots L} P(K_i)$

Как видно, минимум индекса неоднородности достигется при принадлежности всех объектов выборки к одному классу.
Далее алгоритм рекурсивно применяется к потомкам вершины, только в качестве выборки служит один из элементов оптимального разбиения. Построение дерева прекращается когда будет достигнут один из критериев останова:

- Все объекты в вершине имеют один класс
- Достижение максимальной глубины дерева
- Достижение ограничения на минимальное количество объектов в вершине
- Достижение максимального числа листьев

- Слишком малое уменьшение неопределённости.

При использовании первого критерия останова будет построено дерево, которое абсолютно точно классифицирует объекты тренировочной выборки, однако оно может оказаться очень глубоким и совершенно не отражать реальных зависимостей, в результате чего на тестовой выборке покажет плохой результат. Такая систуация называется переобучением (overfitting). Избежать переобучения позволяет подрезка деревьев.
Применим построенное дерево к тестовой выборке и рассмотрим вершину t. Если до t не дошёл ни один объект выборки, то такая вершина неинформативная, и её можно заменить на лист с меткой, соответствующей самому частому классу в поддереве по тренировочной выборке. Если же до t дошли объекты тестовой выборки, то надо вычислить число ошибок на ней и на каждом из её поддеревьев. Если какое-либо из поддеревьев совершает меньше ошибок, то следует заменить t на это поддерево, а остальные отсечь.

Решающие деревья также можно применять для регрессии, только вместо меток классов в лисьях будут храниться значения целевой переменной.

Ансамблирование

Пусть у нас есть семейство *слабых класссификаторов* (тех, у которых вероятность правильного ответа немного превышает случайную). Тогда если усреднить их ответы на произвольном объекте, то точность увеличится по сравнению с одиночным классификатором. А именно, если вероятность правильного ответа одного дерева $P_i > 0.5$, то вероятность ошибки ансамбля равна $\prod_{i=1}^{n}(1-P_i)$ и она уменьшается по мере роста числа деревьев. Построение таких ансамблей оказалось чрезвычайно эффективным и значительно превосходящим многие другие методы ML.

Бэггинг [9]

Одна из ранних версий ансамблей, придуманная Лео Брейманом в 1994 году. В основе бэггинга лежит процедура бутстрепа - выборки с повторениями. Из множества исходных данных отбирается несколько подвыборок, размер которых совпадает с размером исходного множества. Так как отбор происходит случайным образом, то некоторые объекты могут попасть в одну и ту же выборку несколько раз, а в некоторые не попасть вовсе.

Если размер выборки равен n, каждый объект имеет шанс $\frac{1}{n}$ попасть в неё. Вероятность того, что он не попадёт в выборку равна $(1-\frac{1}{n})^n \to \frac{1}{e} \approx 0.36$ при $n \to \infty$
Таким образом каждая подвыборка имеет в среднем $\frac{2}{3}$ уникальных объектов.

Затем на каждой из полученных выборок $X_1, X_2, \ldots, X_l$ обучается отдельный классификатор, который затем будет участвовать в голосовании для определения класса.
Следует отметить, что на больших данных, характерных для машинного обучения, метод не гарантирует отсутствия корреляции между деревьями. Поэтому он получил дальнейнее развитие в более поздних статьях Лео Бреймана.

Случайный лес [10]

Модель является усовершенстованием бэггинга. Она также состоит из бутстрепа и построении деревьев по подвыборкам с повторениями, только в этот раз для каждого дерева установлено ограничиние в k признаков из n исходных, по которым это дерево будет обучаться. Затем голосованием деревьев объект относится к самому популярному классу. Случайный лес, как и бэггинг хорошо распараллеливаются, позволяя быстро получать модели с хорошей точностью.

Out-of-bag оценка

Поскольку тренировочные данные используются неравномерно, при проверке адекватности ансамбля нет необходимости использовать отдельный набор тестовых данных или кросс-валидацию. Можно оценить ошибку отдельного дерева по 36% данных, которые в неё не попали. Это даёт нам оценку на весь ансамбль.

AdaBoost [11]

Алгоритм построения сильного классификатора, предложенный Йоавом Фройндом и Робертом Шапиром в 1996 году. Алгоритм последовательно стоит взвешенный ансамль из деревьев глубины 1 (т.н. "пней"), придавая особое значение тем объектам, которые плохо распознаются предыдущими версиями ансамбля. Рассмотрим пример работы алгоритма в задаче бинарной классификации: Пусть $(x_1, y_1), \ldots, (x_k, y_k)$ - тренировочная выборка, где метка класса принимает 2 значения: $y \in \{-1, 1\}$.
Первоначально все объекты выборки имеют одинаковый вес:
$w_0(x_i) = \frac{1}{m}$
Алгоритм состоит из T иттераций. На t иттерации:

- Строится пень $h_t(x)$, минимизирующий взвешенную ошибку выборки ε.
- Вычисляется $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}$ - значимость t классификатора. При $\varepsilon \to 0.5- \ \alpha_t \to 0$
- Веса объектов обновляются и нормализуются, чтобы получить распределение вероятностей. $w'_{t+1}(i) = w_t(i) e^{-\alpha_t y_i h_t(x_1)}$
  $s = \sum_{i=1}^{n} w'_{t+1}(i), \ \ w_{t+1}(i) = \frac{w'_{t+1}(i)}{s}$
  При этом вес правильно классифицированных объектов уменьшается по отношению к весу неправильно классифицированных.

В результате выполнения алгоритма получается взвешенный ансамбль $f(x) = sign\left(\sum_{i=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost также применим к задачам множественной классификации, регрессии и ранжирования, однако в отличии от бэггинга и случайного леса, не распараллеливается.

Градиентный бустинг [12]

Как и AdaBoost, градиентный бустинг строит сильную модель последовательным добавлением более слабых моделей, обычно деревьев решений.

$(x_1, y_1), \ldots, (x_k, y_k)$ - тренировочная выборка.

Модель будем строить в виде $f(x) = \sum_{t=1}^{T} h_t(x)$, где $h_t(x)$ - дерево решений небольшой глубины.

Для получения качественного алгоритма необходимо минимизировать функцию потерь: $L(y, z)$, где y - правильный ответ, z - прогноз классификатора.

Для регрессии обычно используется среднеквадратическая ошибка: $L(y, f(x)) = (y - f(x))^2$, для классификации - сигмоидная функция потерь: $L(y, f(x)) = 2(1 + e^{yf(x)})^{-1}$.

Важно, чтобы $L(y, z)$ была дифференцируемой.

Построим первый базовый алгоритм $h_1(x)$, который не обязан быть точным. Для регрессии подойдёт среднее значение целевой переменной, а для классификации - мажоритарный класс выборки.

Пусть мы уже построили композицию t алгоритмов $f(x) = \sum_{i=1}^{t-1} h_i$, и хотим улучшить ответ алгоритма, то есть уменьшить суммарное значение функции потерь на всех объектах: $\varepsilon(y, z) = \sum_{i=1}^{k} L(y_i, z)$. Направление наискорейшего убывания $\varepsilon(y, z)$ определяется её антиградиентом. Значит, если мы прибавим к композиции ещё один алгоритм, принимающий значения $h_t(x_i) = aL'_{z_i}(y_i, z_i)$, то ответ несколько улучшится. При этом константа a задаёт величину шага спуска. Если a будет слишком большим, то алгоритм может проскочить локальный минимум. Построение прекращается когда функция потерь достигает своего минимума или величина очередного спуска достаточно мала.

Нейронные сети

Основной принцип модели заимствован у природы. Искусственные нейронные сети (ANN или просто NN), представляют собой орграф, в каждой вершине которого находится искусственный нейрон, а ребра моделируют связи между ними. Нейрон устроен достаточно просто. Он вычисляет взвешенную сумму величин полученных со входящих рёбер, пропускает эту сумму через нелинейную функцию активации и передаёт следующим нейронам. Часть нейронов помечены как входные - на них подаётся информация, а часть как выходные - с них считывается прогноз. В процессе обучения веса ребёр сети меняются, и за счёт этого ANN способны симулировать поведение очень сложных отображений. Обучаются нейронные сети методом обратного распространения ошибки[13].

Перцептрон [14]

В этой архитектуре нейроны сгруппированы в несколько последовательных слоёв. Внутри слоя связей нет, но каждый нейрон одного слоя связан с каждым нейроном следующего слоя. Кроме того первому и последнему слою выделена отдельная роль: нейроны первого слоя считывает сигналы из внешнего мира и не имеют входящих рёбер, а нейроны последнего выдают прогноз и не имеют исходящих рёбер. Все остальные слои являются скрытыми, и их число обычно невелико (1-3). Перцептроны используются для прогнозирования, распознавания образов и управления агентами, но круг решаемых ими задач ограничен. Так, например, Минский показал[15], что перцептроны небольшой глубины неспособны к распознаванию повёрнутых или растянутых объектов. Такие задачи могут возникать при распознавании текста вне зависимости от его ориентации в поле зрения сенсоров.

Глубокое обучение [16]

Наращивание вычислительных мощностей, в первую очередь графических процессоров Nvidia, и накопление больших датасетов позволило к началу 2010 годов обучать очень глубокие нейронные сети с сотнями слоёв и миллиардами связей. Это позволило вывести на качественно новый уровень распознавание образов и лиц, синтез и распознавание речи. Появилось множество предобученных архитектур, применимых практически в любой задаче, связанной с распознаванием.

Свёрточные нейронные сети

CNN состоит из серии чередующихся слоёв свёртки, объединения, слоёв активации и полносвязных слоёв. Выходной слой генерирует вывод с распределением вероятности для каждого ответа. Преимуществом CNN над полновесными НС является гораздо меньшее количество весов, а значит и более высокая скорость обучения при схожем уровне абстракции.

Реккурентные нейронные сети

RNN хорошо приспособлены к обработке протяжённых во времени сигналов за счёт обратных связей между слоями и внутренней памяти, позволяющей решать задачи в контексте последовательностей.

Генетические алгоритмы [17]

Генетические алгоритмы используются в задачах оптимизации и моделирования и заимствуют идеи из эволюционного процесса живых организмов - в их основе лежат мутации, скрещивание и естественный отбор.

Решением задачи оптимизации является особь с наилучшим значением функции приспособленности (fitness). Особь описывается своими генами, обычно представляемыми в виде вектора фиксированной длины.

В качестве первого приближения случайным образом генерируется популяция размера n (не обязательно конкурентоспособная, но достаточно генетически разнообразная), а затем начинается иттеративный эволюционный процесс.

На каждом шаге алгоритма поочерёдно применяются следующие операторы:

Оператор селекции

Аналог естественного отбора в природе. В простейшем случае из полуляции выбирают k наиболее приспособленных особей, чье потомство войдёт в следующее поколение. Однако такой подход может привести к ситуации, когда через несколько поколений весь генотип будет происходить от одного предка и вся популяция застрянет около локального оптимума. Избежать проблемы генетического однообразия позволяет алгоритм NSGA-II:
Пул особей для размножения формируется не из наиболее приспособленых, из наиболее генетически редких, то есть из тех, евклидово расстояние между которыми велико. Затем к

отобранным особям применяется оператор скрещивания, и только потом из полученных $2n$ особей выбираются наиболее приспособленные.

Оператор скрещивания

Обычно скрещивание происходит над двумя особями $x = (x_1, x_2, \ldots, x_l)$ и $y = (y_1, y_2, \ldots, y_l)$ и образует двух потомков. В зависимости от реализации, оно может быть

- одноточечным
  (потомки будут иметь вид $a = (x_1, \ldots x_k, y_{k+1}, \ldots, y_l)$ и $b = (y_1, \ldots y_k, x_{k+1}, \ldots, x_l)$
  точка $k \in 1 \ldots l$ выбирается случайно)
- двухточечным
  $a = (x_1, \ldots x_k, y_{k+1}, \ldots, y_m, x_{m+1}, \ldots x_l)$ и $b = (y_1, \ldots, y_k, x_{k+1}, \ldots, x_m, y_{m+1}, \ldots, y_l)$
- по произвольной битовой маске, где i ген отнесётся к первому потомку, если i бит маски 1, и ко второму в противном случае.

В некоторых реализациях возможно скрещивание с использованием трёх и более родителей.

Оператор мутации

Применимо к конкретной особи, оператор меняет один или несколько генов на некоторую величину, позволяя добиться бóльшего генетического разнообразия или выйти из локального экстремума.

Эволюция останавливается, когда будет достигнут один из критерий останова:

- Нахождение глобального или локального решения
- Достижение максимального числа поколений
- Превышение времени работы алгоритма

Результаты, достигнутые в области

Большинство из используемых на сегодняшний день методов машинного обучения были разработаты ещё в 60-е годы XX века, но повсеместное их использование началось во втором десятилетии XXI века, когда человечество накопило большое количество информации, а вычислительные мощности стали достаточно дешёвыми. Ниже приведены наиболее существенными существенные достижения в области за последние 10 лет:

Распознавание речи

Согласно исследованию Microsoft[18], проведённому на датасете NIST 2000, в 2016 году системы распознавания речи достигли человеческого уровня. В разговорах малознакомых людей на заданную тему точность распознавания составила 5.8%, а в неформальных разговорах друзей

и родственников - 11%, что на 0.1% и 0.3% больше точности профессиональных стенографистов. Добиться таких результатов помогла комбинация CNN и RNN с долгой краткосрочной памятью.

Синтез речи

В решении обратной задачи также участвовали свёрточные нейросети. В 2016 году Deep Mind опубликовал[19] статью о WaveNet - CNN, генерирующей аудиозаписи, в том числе и речь по последовательности фонетических характеристик. WaveNet способна имитировать голоса различных людей, акцент и эмоции. Средняя субъективная оценка качества синтезированной речи составляет 4.21 из 5 для английского языка и 4.08 для китайского. Для сравнения: у носителей языка эта оценка составляет 4.55 и 4.21 соответственно.

Распознавание образов

Одна из ранних глубоких CNN, AlexNet, в 2012 году дебютировала в конкурсе по распознаванию образов ImageNet, значительно превзойдя предыдущий рекорд. Ошибка модели составила 16%, и в дальнейших соревнованиях первое место неизменно занимали именно CNN, уменьшив оценку до нескольких процентов. А согласно исследованию 2015 года[20], системы распознавания образов превзошли человеческий уровень.

Медицинская диагностика

Потеницал применения AI в медицине довольно велик, о чём свидетельствует растущий поток инвестиций, однако его использование сопряжено с рядом трудностей:

- Плохая интерпретируемость некоторых моделей
- Трудность сертификации
- Отсутствие доверия у населения

В связи с этим большинство решений позиционирует себя не как полную замену врачу, а как систему поддержки принятия решений. К числу наиболее заметных работ последних лет можно отнести работы Стенфордского университета[21] по классификации рентгенограмм пациентов. Модель на основе CNN способна диагностировать одно из 14 заболеваний, а пневмонию она определяет лучше врачей-рентгенологов.

Обнаружение спама

В задачах фильтрации спама хорошо зарекомендовал себя Байесовский классификатор[22]. При обучении классификатора высчитывается вероятность того, что письмо с данным словом - спам, а при классификации нового письма, вероятность, что оно окажется спамом оценивается через вероятности составляющих его слов. Метод не требует больших вычислительных ресурсов, способен дообучаться и показывает 95-97% точность, за счет чего лежит в основе большинства современных спам-фильтров.

Улучшение качества изображений [23]

Набор техник, получивший название Super Resolution, позволяет конструировать изображение высокого разрешения из низкого, исправляя шум и размытость, однако, поскольку восстановление происходит по одному кадру, медод не привносит дополнительной детализации, а только делает изображение более приятным для человека.

Улучшение качества видео [24]

Методы video super resolution для реконструкции используют информацию с предыдущих и последующих кадров, и за счёт этого наблюдается существенный прирост детализации. Но даже на современных GPU, полная обработка 1 минуты видео может занимать недели.
В противовес реальному video SR существуют методики улучшения видео в режиме реального времени, например преобразование ТВ сигнала низкого качества в HD. Однако реконструкция происходит по одному кадру, и по смыслу такое восстановление ближе к image super resolution.

Голосовые ассистенты [25]

Современные голосовые ассистенты, встроенные в операционную систему или программное обеспечение наиболее близко подобрались с понятию искусственного интеллекта. Они решают срезу несколько задач: speech-to-text преобразование, семантический анализ текста, запросы к сервисам, преобразование ответа обратно в речь. По заявляниям Яндекса[26], спрос на голосовых помощников постоянно растёт, и по состоянию на май 2018 года, голосовой ассистент «Алиса» установлена на 53% российских сматрфонах и доступна в навигаторах 20 млн автомобилей.

Список литературы

[1] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[2] В.В.Степаненко И.И.Холод А.А.Барсегян, М.С.Куприянов. *Методы и модели анализа данных, OLAP и Data Mining.* БХВ-Петербург, 2004.

[3] Н.В.Александрова. *История математических терминов, понятий и обозначений.* Издательство ЛКИ, 2008.

[4] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[5] Hastie T., Tibshirani R., and Friedman J. *The Elements of Statistical Learning.* Springer, 2001.

[6] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug 2004.

[7] Vladimir Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural computation*, 12(9):2013–2036, 2000.

[8] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] Leo Breiman. Random forests. *Machine Learning*, 45(1):pp 5–32, 2001.

[11] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[12] Robert E Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[13] Александр Иванович Галушкин. *Синтез многослойных систем распознавания образов.* Энергия, 1974.

[14] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[15] M Минский and C Пейперт. Персептроны. 1971.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:pp 436–444, 2015.

[17] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[18] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. Achieving human parity in conversational speech recognition, 2016.

[19] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[20] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[21] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning, 2017.

[22] Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, George Paliouras, and Constantine D Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*, 2000.

[23] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[24] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016.

[25] Gustavo López, Luis Quesada, and Luis A. Guerrero. Alexa vs. siri vs. cortana vs. google assistant: A comparison of speech-based natural user interfaces. In Isabel L. Nunes, editor, *Advances in Human Factors and Systems Interaction*, pages 241–250, Cham, 2018. Springer International Publishing.

[26] Конференция Яндекса. https://events.yandex.ru/events/b-konf/16-july-2018/. 2018-07-16.