

Машинное обучение

1. Чем отличаются модели машинного обучения от других пусть даже и очень сложных систем и моделей? Приведите примеры задач машинного обучения и способов их решения.

Методы машинного обучения решают задачу не напрямую, а после обучения в процессе решения множества сходных задач. Они не могут выявить точную закономерность и дают приблизительный ответ.

https://ru.wikipedia.org/wiki/Машинное_обучение

Классические задачи, решаемые с помощью машинного обучения

- [Классификация](#) как правило, выполняется с помощью [обучения с учителем](#) на этапе собственно обучения.
- [Кластеризация](#) как правило, выполняется с помощью [обучения без учителя](#)
- [Регрессия](#) как правило, выполняется с помощью [обучения с учителем](#) на этапе тестирования, является частным случаем [задач прогнозирования](#).
- Понижение размерности данных и их визуализация выполняется с помощью [обучения без учителя](#)
- Восстановление плотности распределения вероятности по набору данных
- Одноклассовая классификация и выявление новизны
- Построение ранговых зависимостей

2. Чем отличаются алгоритмы с учителем (supervised) и без учителя (unsupervised)? Приведите примеры подобных алгоритмов и задач, которые они помогают решить.

При обучении с учителем системе даётся набор пар «ситуация — решение».

Применяются для задач регрессии и классификации.

Примеры методов:

- Метод обратного распространения ошибки (back propagation)
- Метод упругого распространения (resilient propagation)
- Метод опорных векторов

Примеры задач:

- Принятие решения о выдаче или невыдаче кредита
- Расчёт заработка разносчика еды в определённый день

При обучении без учителя системе даётся только набор ситуаций. Применяются в задачах кластеризации.

Примеры методов:

- Метод ближайших соседей/метод k-средних (k-means)
- Основанная на плотности пространственная кластеризация для приложений с шумами (DBSCAN)

Примеры задач:

- Выделение групп слов с похожей семантикой
- Выделение границ в изображении

3. Как работает метод градиентного спуска и как он используется в алгоритмах машинного обучения?

Метод градиентного спуска находит градиент функции стоимости в текущей точке и малыми шагами движется в сторону градиента, чтобы найти минимум функции и получить наименьшее значение ошибки. Положен в основу метода back propagation.
<https://craftappmobile.com/gradient-descent-tutorial/>

4. Как работает метод наименьших квадратов и какие задачи он позволяет решать?

Метод наименьших квадратов применяется для нахождения линейной регрессии. Он минимизирует среднеквадратичную ошибку между реальным и прогнозируемым значениями. Позволяет решать любые оптимизационные задачи (регрессия, классификация).

5. Почему метод наименьших квадратов не лучшим образом подходит для задач классификации? Каким образом логистическая регрессия решает эту проблему?

Метод наименьших квадратов реализуется через нормальное уравнение. Для решения нормального уравнения требуется найти обратную матрицу. Само нахождение обратной матрицы является дорогостоящей операцией, т. к. её сложность n^3 . К тому же её не всегда возможно найти.

Логистическая регрессия - это статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём подгонки данных к логистической кривой. В зависимости от выбранной логистической функции и выходных данных позволяет определить принадлежность объекта к какому то классу.

6. Какими способами можно работать с категориальными (номинативными) переменными?

- Label encoding переводит значение категориальной переменной в вещественное значение.
- One-hot (dummy) encoding предполагает создание по одному бинарному признаку на каждое значение категориальной переменной, все из которых будут равны 0, за исключением одного.

[Про label + onehot encoding](#)

7. Что такое целевая функция (функция стоимости, cost function)?

Приведите примеры таких функций для различных алгоритмов машинного обучения.

Целевая функция - функция, которая сравнивает действительное значение целевого признака и предсказанное значение.

Пример средней квадратичной ошибки:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

Root MSE

$$\sqrt{\frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + \dots + (i_n - a_n)^2}{n}}$$

Arctan

$$\frac{\arctan^2(i_1 - a_1) + \dots + \arctan^2(i_n - a_n)}{n}$$

8. Почему критерий точности (accuracy) не всегда идеально подходит для оценки качества модели? Какие еще критерии можно использовать и как они помогают в оценке?

При обучении на тестовой выборке мы можем учесть не все признаки изучаемого объекта. Причины тому: малое количество данных (обучающая выборка не полностью описывает объект), неправильно выбранная модель (функция).

Кроме критерия точности, можно также использовать коэффициент корреляции и коэффициент детерминации. Про них здесь:

<http://www.prognoz.ru/blog/platform/kriterii-otsenki-modeli/>

9. Что такое переобучение, почему это плохо и какими методами с ним можно бороться?

При переобучении (overfitting) построенная модель теряет способность обобщать информацию и просто запоминает ответы, подстраиваясь под конкретную выборку, а не под генеральную совокупность. В результате переобучения падает точность ответов. Методы предотвращения переобучения:

- **Перекры́стная проверка (cross-validation)** — данные разбиваются на n частей, из которых $n - 1$ используются для обучения, а оставшаяся — для тестирования. Этот процесс повторяется n раз так, чтобы все части были по разу использованы для тестирования.
- **Регуляризация** — введение «штрафов» за слишком сложную структуру (например, за большие значения параметров).
- **Ранняя остановка** — прерывание обучения, если за заданное количество эпох ошибка не уменьшается.
- **Исключение/прореживание (dropout)** — исключение части нейронов из сети с тем, чтобы они перестали влиять на результаты её работы.

10. Опишите принципы работы деревьев принятия решений.

Дерево принятия решений состоит из ребер, узлов и листьев. Узлы представляют собой какой либо признак, по которому данным можно разделить (один из предикатов), листья представляют собой варианты решений, в ребрах содержатся значения, которые признаки из узлов могут принимать. Обучение деревьев может происходить различными способами, но использование их для классификации всегда одинаковое - спуск по дереву и выбор нужного ребра соответственно значению признака.

Область применения дерева решений в настоящее время широка, но все задачи, решаемые этим аппаратом могут быть объединены в следующие три класса:

- **Описание данных:** деревья решений позволяют хранить информацию о данных в компактной форме, при этом они содержат точное описание объектов.
- **Классификация:** деревья решений отлично справляются с задачами классификации, т.е. отнесения объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.
- **Регрессия:** если целевая переменная имеет непрерывные значения, деревья решений позволяют установить зависимость целевой переменной от независимых(входных) переменных. Например, к этому классу относятся задачи численного прогнозирования (предсказания значений целевой переменной).

11. Как работает k-NN и какие задачи он позволяет решать?

kNN расшифровывается как k Nearest Neighbor или k Ближайших Соседей — это один из самых простых алгоритмов классификации, также иногда используемый в задачах регрессии.

Рассмотрим задачу классификации.

Пусть имеется m наблюдений, каждому из которых соответствует запись в таблице. Все записи принадлежат какому-либо классу. Необходимо определить класс для новой записи.

На первом шаге алгоритма следует задать число k – количество ближайших соседей. Если принять $k = 1$, то алгоритм потеряет обобщающую способность (то есть способность выдавать правильный результат для данных, не встречавшихся ранее в алгоритме) так как новой записи будет присвоен класс самой близкой к ней. Если установить слишком большое значение, то многие локальные особенности не будут выявлены. Проблему выбора оптимального значения параметра k называют «bias-variance tradeoff», т.е. «компромисс между «выбросами» и дисперсией». На практике чаще всего полагают $k = \lfloor \sqrt{N} \rfloor$.

Согласно методу k-NN мы отнесем его к тому классу, к которому принадлежит большинство из k его ближайших соседей. Расстояние между объектами будем

понимать в смысле Евклидовой нормы $D = \sqrt{\sum_i^n (x_i - y_i)^2}$, т.е. расстояние между объектами с координатами (x_1, y_1) и (x_2, y_2) равно $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

При нахождении расстояния иногда учитывают значимость атрибутов. Она определяется экспертом или аналитиком субъективно на основе накопленного опыта. В таком случае при нахождении расстояния каждый i -ый квадрат разности в сумме умножается на коэффициент Z_i . Например, если атрибут А в три раза важнее атрибута В ($Z_A = 3$, $Z_B = 1$), то расстояние будет находиться следующим образом:

$$D = \sqrt{3(x_a - y_a)^2 + (x_b - y_b)^2}$$

Подобный прием называют растяжением осей (stretching the axes), что позволяет снизить ошибку классификации.

На практике различные признаки могут иметь разные единицы измерения и разные шкалы, что может существенно исказить реальное расстояние между объектами. Для решения этой проблемы перед применением метода k-NN производят так называемую нормализацию (или масштабирование) данных (англ. scaling).

Нормализация с помощью стандартного отклонения:

$X = (x_i - \bar{x})/s$, где s — стандартное отклонение, \bar{x} — среднее.

$x = (x_i - x_{min}) / (x_{max} - x_{min})$ — переход от абсолютных значений признаков к относительным. Преимущество новых переменных состоит в том, что они принимают значения от 0 до 1 (или, если перейти к процентному выражению, то от 0 до 100).

Алгоритм k-ближайших соседей имеет широкое применение. Например:

1. **Обнаружение мошенничества.** Новые случаи мошенничества могут быть похожи на те, которые происходили когда-то в прошлом. Алгоритм KNN может распознать их для дальнейшего рассмотрения.
2. Предсказание отклика клиентов. Можно определить отклик новых клиентов по данным из прошлого.
3. Медицина. Алгоритм может классифицировать пациентов по разным показателям, основываясь на данных прошедших периодов.
4. Прочие задачи, требующие классификацию.

В заключение отметим достоинства и недостатки алгоритма KNN.

Перечислим положительные особенности.

1. Алгоритм устойчив к аномальным выбросам, так как вероятность попадания такой записи в число k-ближайших соседей мала. Если же это произошло, то влияние на голосование (особенно взвешенное) (при $k > 2$) также, скорее всего, будет незначительным, и, следовательно, малым будет и влияние на итог классификации.
2. Программная реализация алгоритма относительно проста.
3. Результат работы алгоритма легко поддаётся интерпретации. Экспертам в различных областях вполне понятна логика работы алгоритма, основанная на нахождении схожих объектов.
4. Возможность модификации алгоритма, путём использования наиболее подходящих функций сочетания и метрик позволяет подстроить алгоритм под конкретную задачу.

Алгоритм KNN обладает и рядом недостатков. Во-первых, набор данных, используемый для алгоритма, должен быть репрезентативным. Во-вторых, модель нельзя "отделить" от данных: для классификации нового примера нужно использовать все примеры. Эта особенность сильно ограничивает использование алгоритма.

12. Как поступать если в задаче классификации мы имеем дело со множеством классов (> 2)?

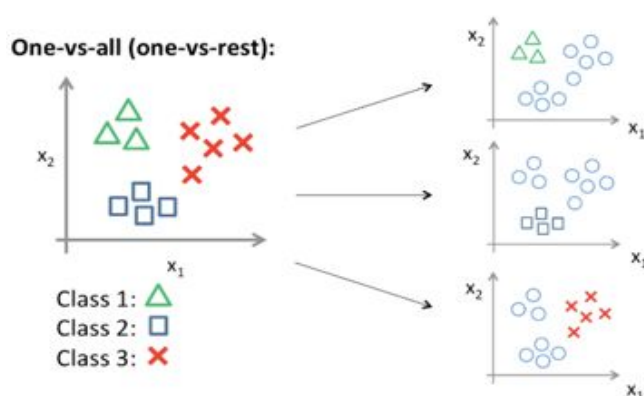
One-Hot Encoding

Предположим, что некоторый признак может принимать 10 разных значений. В этом случае One Hot Encoding подразумевает создание 10 признаков, все из которых равны нулю за исключением одного. На позицию, соответствующую численному значению признака мы помещаем 1.

В то время как некоторые алгоритмы классификации допускают использование более двух классов по своему замыслу, другие ограничивают возможные результаты одним из двух значений (двоичной или двухклассовой моделью). Однако даже двоичные алгоритмы классификации могут быть адаптированы для задач классификации нескольких классов используя различные стратегии.

Этот модуль реализует метод «один против всех», в котором двоичная модель создается для каждого из нескольких выходных классов. Каждая из этих бинарных моделей для отдельных классов оценивается с точки зрения ее дополнения (всех других классов в модели), как если бы это была проблема бинарной классификации. Затем прогнозирование выполняется путем запуска этих двоичных классификаторов и выбора прогноза с наибольшим доверительным счетом.

По сути, создается множество отдельных моделей, а затем результаты объединяются для создания единой модели, которая прогнозирует все классы. Таким образом, любой двоичный классификатор может использоваться в качестве основы для модели «один против всех».

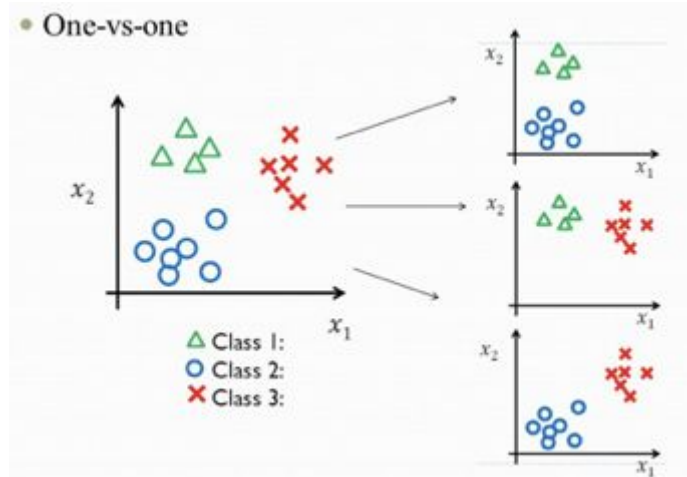


In the one-vs.-one (OvO) reduction, one trains $K(K - 1) / 2$ binary classifiers for a K -way multiclass problem; each receives the samples of a pair of classes from the original training set, and must learn to distinguish these two classes. At prediction time, a voting scheme is applied: all $K(K - 1) / 2$ classifiers are applied to an unseen sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier.

Like OvR, OvO suffers from ambiguities in that some regions of its input space may receive the same number of votes.

Долбоёбский/машинный перевод:

В сокращении « один против одного» (OvO) обучают двоичные классификаторы $K (K - 1) / 2$ для задачи мультикласса на K - путях; каждый получает образцы пары классов из исходного учебного набора и должен научиться различать эти два класса. Во время прогнозирования применяется схема голосования: все классификаторы $K (K - 1) / 2$ применяются к невидимой выборке, а класс, получивший наибольшее число прогнозов "+1", прогнозируется комбинированным классификатором. Как и OvA, OvO страдает от двусмысленности в том, что некоторые области его входного пространства могут получать одинаковое количество голосов.



13. Каким образом можно применять линейные методы в случае нелинейных зависимостей?

https://ru.wikipedia.org/wiki/Нелинейная_регрессия#Линеаризация

14. Что такое задача кластеризации, какие есть способы ее решения и как они устроены?

Интуитивная постановка задачи кластеризации довольно проста и представляет из себя наше желание сказать: "Вот тут у меня насыпаны точки. Я вижу, что они сваливаются в какие-то кучки вместе. Было бы круто иметь возможность эти точки относить к кучкам и в случае появления новой точки на плоскости говорить, в какую кучку она падает." (Без учителя)

K-means

Алгоритм К-средних, наверное, самый популярный и простой алгоритм кластеризации и очень легко представляется в виде простого псевдокода:

1. Выбрать количество кластеров, которое нам кажется оптимальным для наших данных.
2. Высыпать случайным образом в пространство наших данных точек (центроидов).
3. Для каждой точки нашего набора данных посчитать, к какому центроиду она ближе.
4. Переместить каждый центроид в центр выборки, которую мы отнесли к этому центроиду.
5. Повторять последние два шага фиксированное число раз, либо до тех пор пока центроиды не "сойдутся" (обычно это значит, что их смещение относительно предыдущего положения не превышает какого-то заранее заданного небольшого значения).

DBSCAN (Density-based spatial clustering of applications with noise, плотностной алгоритм пространственной кластеризации с присутствием шума), как следует из названия, оперирует плотностью данных.

Интуитивное объяснение

В гигантском зале толпа людей справляет чей-то день рождения. Кто-то слоняется один, но большинство — с товарищами. Некоторые компании просто толпятся гурьбой, некоторые — водят хороводы или танцуют ламбаду.

Мы хотим разбить людей в зале на группы.

Будем говорить, что рядом с некоторым человеком собралась толпа, если близко к нему стоят несколько других человек. Ага, сразу видно, что нужно задать два параметра. Что значит «близко»? Возьмём какое-нибудь интуитивно понятное расстояние. Скажем, если люди могут дотронуться до голов друг друга, то они находятся близко. Около метра. Теперь, сколько именно «несколько других человек»? Допустим, три человека. Двое могут гулять и просто так, но третий — определённо лишний.

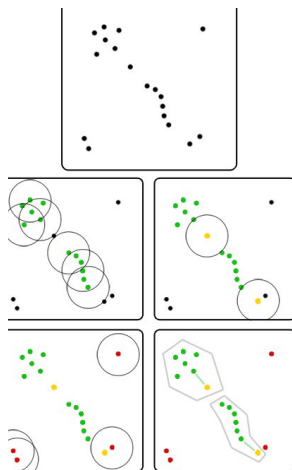
Пусть каждый подсчитает, сколько человек стоят в радиусе метра от него. Все, у кого есть хотя бы три соседа, берут в руки зелёные флажки. Теперь они коренные элементы, именно они формируют группы.

Обратимся к людям, у которых меньше трёх соседей. Выберем тех, у которых по крайней мере один сосед держит зелёный флаг, и вручим им жёлтые флаги. Скажем, что они находятся на границе групп.

Остались одиночки, у которых мало того что нет трёх соседей, так ещё и ни один из них не держит зелёный флаг. Раздадим им красные флаги. Будем считать, что они не принадлежат ни одной группе.

Таким образом, если от одного человека до другого можно создать цепочку «зелёных» людей, то эти два человека принадлежат одной группе. Очевидно, что все подобные скопища разделены либо пустым пространством либо людьми с жёлтыми флагами. Можно их пронумеровать: каждый в группе №1 может достичь по цепочке рук каждого другого в группе №1, но никого в №2, №3 и так далее. То же для остальных групп.

Если рядом с человеком с жёлтым флажком есть только один «зелёный» сосед, то он будет принадлежать той группе, к которой принадлежит его сосед. Если таких соседей несколько, и у них разные группы, то придётся выбирать. Тут можно воспользоваться разными методами — посмотреть, кто из соседей ближайший, например. Придётся как-то обходить краевые случаи, но ничего страшного.



Как правило, нет смысла помечать всех коренных элементов толпы сразу. Раз от каждого коренного элемента группы можно провести цепочку до каждого другого, то всё равно с какого начинать обход — рано или поздно найдёшь всех. Тут лучше подходит итеративный вариант:

1. Подходим к случайному человеку из толпы.
2. Если рядом с ним меньше трёх человек, переносим его в список возможных отшельников и выбираем кого-нибудь другого.
3. Иначе:

- Исключаем его из списка людей, которых надо обойти.
 - Вручаем этому человеку зелёный флажок и создаём новую группу, в которой он пока что единственный обитатель.
 - Обходим всех его соседей. Если его сосед уже в списке потенциальных одиночек или рядом с ним мало других людей, то перед нами край толпы. Для простоты можно сразу пометить его жёлтым флагом, присоединить к группе и продолжить обход. Если сосед тоже оказывается «зелёным», то он не стартует новую группу, а присоединяется к уже созданной; кроме того мы добавляем в список обхода соседей соседа. Повторяем этот пункт, пока список обхода не окажется пуст.
4. Повторяем шаги 1-3, пока так или иначе не обойдём всех людей.
 5. Разбираемся со списком отшельников. Если на шаге 3 мы уже раскидали всех краевых, то в нём остались только выбросы-одиночки — можно сразу закончить. Если нет, то нужно как-нибудь распределить людей, оставшихся в списке.

В целом это очень напоминает обход графа в ширину и выделение компонент связности.

15. В чем сложность работы с текстом и какие существуют для этого способы?

Текстовые данные — это слова, которые вместе несут некоторую информацию. При этом, например, имя и фамилия текстом не являются.

Отдельное слово называется термином (term). Фразы, составленные из слов, называются документами (documents). Совокупность документов называется корпусом (corpus).

При работе с текстом встречаются следующие проблемы:

- Различные формы одного и того же слова, измененного суффиксами (кот - котик) либо окончаниями (близкий - близкая).
- Обилие бессмысленных слов (артикли в английском языке, предлоги и частицы, которые не несут смысловой нагрузки)

Эти задачи решаются при процессе токенизации текста. Первым делом текст разбивается на отдельные слова (пунктуация и case (Aa) игнорируются), затем из этого числа слов удаляются лишние слова (артикли и тп), далее возможно приведение всех слов к сокращенной форме (стемминг, удаление суффиксов, окончаний). После этого происходит векторизация полученного корпуса при помощи нужного способа (самый популярный tf-idf) и можно выполнять нужные операции.

Векторизация текста — это сопоставление чисел с ним.

1. Count vectorizer — подсчитывает вхождение каждого слова в текст.
2. Inverse document frequency (IDF) рассчитывается по формуле: (число всех документов) / (число документов, в которых встречается слово). Также может использоваться логарифм от данного выражения, чтобы разница между часто и редко встречающимися словами была не такой большой.
3. Term frequency (TF) рассчитывается по формуле: (количество вхождений слова) / (количество всех слов).
4. TF-IDF рассчитывается по формуле $TF * IDF$.

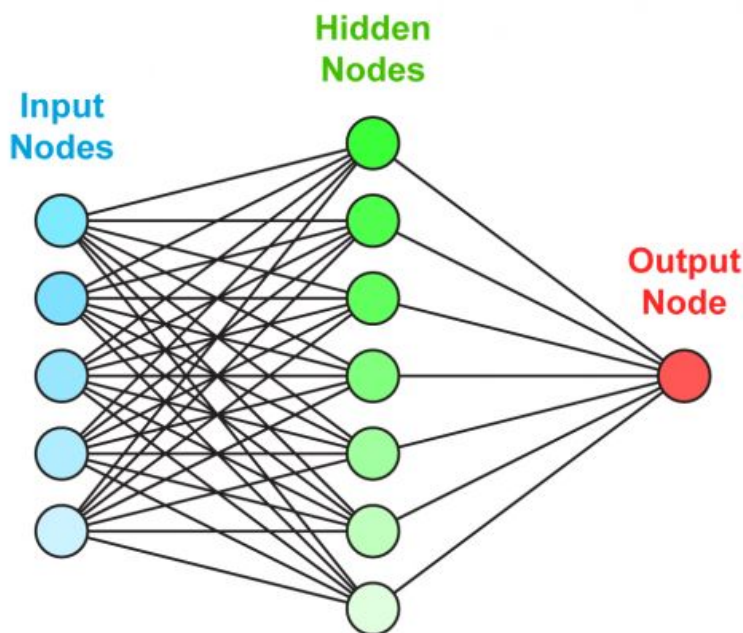
16. Как устроена нейронная сеть и на каких принципах работает?

Нейронная сеть — попытка с помощью математических моделей воспроизвести работу человеческого мозга для создания машин, обладающих искусственным интеллектом.

Искусственная нейронная сеть обычно обучается с учителем. Это означает наличие обучающего набора (датасета), который содержит примеры с истинными значениями: тегами, классами, показателями.

Искусственная нейронная сеть состоит из трех компонентов:

- Входной слой;
- Скрытые (вычислительные) слои;
- Выходной слой.



Обучение нейросетей происходит в два этапа:

- Прямое распространение ошибки;
- Обратное распространение ошибки.

Во время прямого распространения ошибки делается предсказание ответа. При обратном распространении ошибка между фактическим ответом и предсказанным минимизируется.

<https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/>

17. С какими проблемами мы сталкиваемся увеличивая количество слоев нейронной сети? Как можно ослабить эти проблемы?

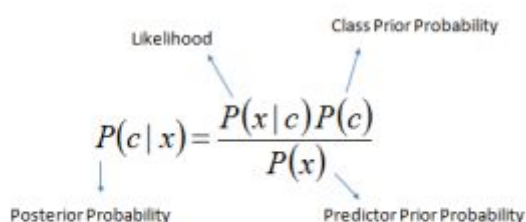
- Проблема затухания градиента (ослабление входного сигнала)
Чем больше слоёв в нейронной сети, тем дольше приходится её обучать.
Чтобы избежать этого, в качестве функций активации можно использовать Rectified Linear Unit (напр., в скрытых слоях) и Softmax (напр., в выходном слое).
- Не хватает input dataset'a (слишком мало вариативности)

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as $f(x)=\max(0,x)$.

18. Опишите принцип работы наивного байесовского классификатора (naive bayes).

Наивный байесовский алгоритм – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков. Другими словами, НБА предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака. Например, фрукт может считаться яблоком, если он красный, круглый и его диаметр составляет порядка 8 сантиметров. Даже если эти признаки зависят друг от друга или от других признаков, в любом случае они вносят независимый вклад в вероятность того, что этот фрукт является яблоком. В связи с таким допущением алгоритм называется «наивным».

Теорема Байеса позволяет рассчитать апостериорную вероятность $P(c|x)$ на основе $P(c)$, $P(x)$ и $P(x|c)$.


$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Принцип работы на примере

<http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/>

Положительные стороны:

Классификация, в том числе многоклассовая, выполняется легко и быстро.

Когда допущение о независимости выполняется, НБА превосходит другие алгоритмы, такие как логистическая регрессия (logistic regression), и при этом требует меньший объем обучающих данных.

НБА лучше работает с категориальными признаками, чем с непрерывными. Для непрерывных признаков предполагается нормальное распределение, что является достаточно сильным допущением.

Отрицательные стороны:

нужны изначальные данные чтобы классифицировать

У нас есть тренировочный набор данных о 1000 фруктах.

Фрукт может быть бананом, апельсином или каким-нибудь другим (это классы).

Фрукт может быть длинным, сладким или желтым (это параметры).

Class	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

Что мы видим в этом тренировочном наборе данных?

Из 500 бананов 400 длинные, 350 сладкие и 450 желтые;

Среди 300 апельсинов нет ни одного длинного, но оказалось 150 сладких и 300 желтых.

Из оставшихся 200 фруктов 100 оказались длинными, 150 сладкими и 50 желтыми.

Если мы получим только параметры – длину, сладость и цвет фрукта (не зная его класса), то сможем вычислить вероятность того, что фрукт окажется бананом, апельсином или чем-то другим.

Предположим, что неизвестный фрукт длинный, сладкий и желтый.

Для вычисления вероятности нужно проделать 4 простых шага:

Шаг 1: Чтобы вычислить вероятность того, что неизвестный фрукт – это банан, давайте сначала решим, похож ли этот фрукт на банан. Вот как вычисляется вероятность класса «Банан» на основании параметров «длинный», «сладкий», «желтый»:

$P(\text{Banana}|\text{Long, Sweet, Yellow})$

Выглядит точно так же как уравнение, описанное выше.

Шаг 2: Начнем с числителя и подставим все значения в уравнение:

$$P(\text{Long}|\text{Banana}) = 400/500 = 0.8$$

$$P(\text{Sweet}|\text{Banana}) = 350/500 = 0.7$$

$$P(\text{Yellow}|\text{Banana}) = 450/500 = 0.9$$

$$P(\text{Banana}) = 500/1000 = 0.5$$

Перемножив значения (согласно уравнению), мы получим:

$$0.8 \times 0.7 \times 0.9 \times 0.5 = 0.252$$

Шаг 3: Проигнорируем знаменатель, поскольку он будет одинаковым для всех последующих вычислений.

Шаг 4: Проделаем те же вычисления для других классов:

$$P(\text{Orange}|\text{Long, Sweet, Yellow}) = 0$$

$$P(\text{Other}|\text{Long, Sweet, Yellow}) = 0.01875$$

Поскольку 0,252 больше, чем 0,01875, то наивный байесовский алгоритм классифицирует этот длинный, сладкий и желтый фрукт как банан.

Требуется ли этот метод обучения или он самообучающийся? Этот метод требует обучения, поскольку алгоритм использует размеченный набор данных для построения таблицы.

Почему стоит использовать наивный байесовский классификатор? Как вы можете заметить в примере выше, алгоритм состоит из простой арифметики. Это простые вычисления: умножение и деление.

Когда частотные таблицы уже вычислены, классификация неизвестного фрукта включает в себя только вычисления вероятностей для всех классов, а затем выбор наибольшей вероятности.

Несмотря на простоту, наивный байесовский алгоритм может быть удивительно точным. Например, было установлено, что он может быть успешно применен для фильтрации спама.

19. Как работает сверточная нейронная сеть (convolutional neural network), какие задачи она помогает решить?

Сверточная нейронная сеть (convolutional neural network) — специальная архитектура искусственных нейронных сетей, нацеленная на эффективное распознавание изображений, входит в состав технологий глубокого обучения (deep learning). Использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определенного набора простых клеток. Таким образом, идея сверточных нейронных сетей заключается в чередовании сверточных слоев (convolution layers) и субдискретизирующих слоев (subsampling layers, слоев подвыборки). Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

Работа сверточной нейронной сети обеспечивается двумя основными элементами:

- Фильтры (filters) (определители признаков)
- Карты признаков (feature maps).

Что конкретно делают СНС? Берётся изображение, пропускается через серию свёрточных, нелинейных слоев, слоев объединения и полносвязных слоёв, и генерируется вывод. Выводом может быть класс или вероятность классов, которые лучше всего описывают изображение.

Подробнее на примере: <https://habr.com/ru/post/309508/>

Свёрточные нейронные сети используются при обработке изображений и текста. Также возможно обрабатывать аудио и видео.

20. Как работает рекуррентная нейронная сеть (recurrent neural network), какие задачи она помогает решить?

В отличие от нейронных сетей прямого распространения, в рекуррентных сетях сигнал распространяется не только от входов к выходам, но и обратно. Это позволяет сохранять в процессе обработки данных информацию, которая затем может быть использована для обработки следующих данных.

В рекуррентных нейронных сетях может быть реализовано простое запоминание определенного количества предыдущих шагов или же более сложное устройство, которое определяет, что стоит запомнить, а что нет.

Рекуррентные нейронные сети используются при обработке последовательностей, то есть когда важен порядок данных, в котором они поступают, и применяются:

- для перевода текстов (машинный перевод, ага);
- для распознавания речи;
- для распознавания эмоций;
- для предсказания следующего кадра видео на основе предыдущих.

Где используют рекуррентные нейросети?

Рекуррентные нейронные сети продемонстрировали большой успех во многих задачах NLP. На этом этапе нужно упомянуть, что наиболее часто используемым типом RNN являются LSTM, которые намного лучше захватывают (хранят) долгосрочные зависимости, чем RNN. Но не волнуйтесь, [LSTM](#) — это, по сути, то же самое, что и RNN, которые мы разберем в этом уроке, у них просто есть другой способ вычисления скрытого состояния. Более подробно мы рассмотрим LSTM в другом посте. Вот некоторые примеры приложений RNN в NLP (без ссылок на исчерпывающий список).

Языковое моделирование и генерация текстов

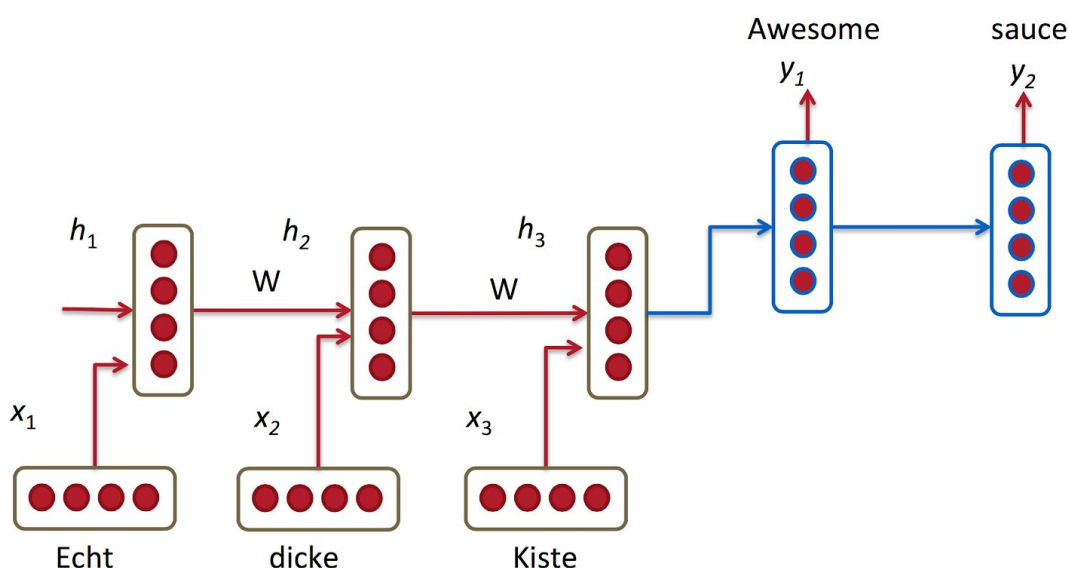
Учитывая последовательность слов, мы хотим предсказать вероятность каждого слова (в словаре). Языковые модели позволяют нам измерить вероятность выбора, что является важным вкладом в машинный перевод (поскольку предложения с большой вероятностью правильны). Побочным эффектом такой способности является возможность генерировать новые тексты путем выбора из выходных вероятностей. Мы можем генерировать и другие [вещи](#), в зависимости от того, что из себя представляют наши данные. В языковом моделировании наш вход обычно представляет последовательность слов (например, закодированных как вектор с одним горячим состоянием (one-hot)), а выход — последовательность предсказанных слов. При обучении [нейронной сети](#), мы подаем на вход следующему слою предыдущий выход $o_t = x_{t+1}$, поскольку хотим, чтобы результат на шаге t был следующим словом.

Исследования по языковому моделированию и генерации текста:

- [Word2Vec: как работать с векторными представлениями слов](#)
- [NLP Architect от Intel: open source библиотека моделей обработки естественного языка](#)

Машинный перевод

Машинный перевод похож на языковое моделирование, поскольку вектор входных параметров представляет собой последовательность слов на исходном языке (например, на немецком). Мы хотим получить последовательность слов на целевом языке (например, на английском). Ключевое различие заключается в том, что мы получим эту последовательность только после того, как увидим все входные параметры, поскольку первое слово переводимого предложения может потребовать информации всей последовательности вводимых слов.



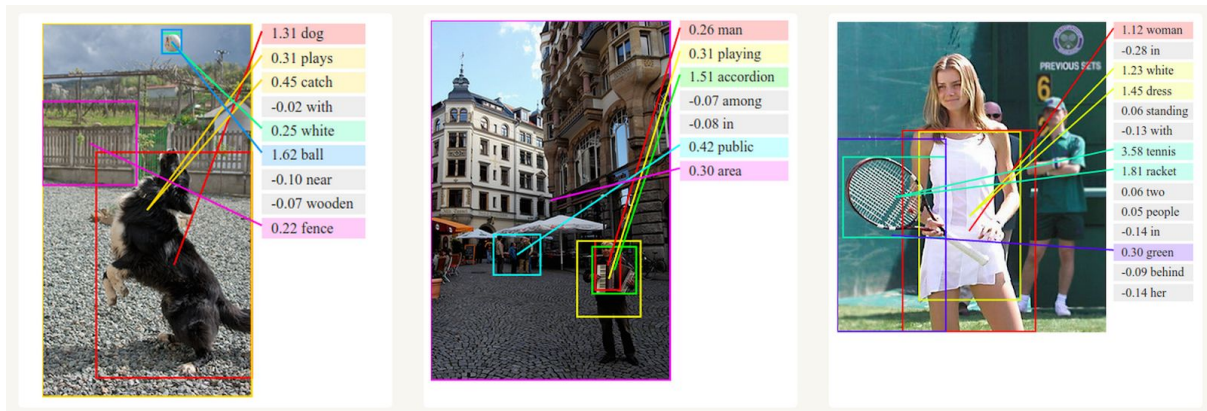
RNN для машинного перевода

Распознавание речи

По входной последовательности акустических сигналов от звуковой волны, мы можем предсказать последовательность фонетических сегментов вместе со своими вероятностями.

Генерация описания изображений

Вместе со сверточными нейронными сетями RNN использовались как часть модели генерации описаний неразмеченных изображений. Удивительно, насколько хорошо они работают. Комбинированная модель совмещает сгенерированные слова с признаками, найденными на изображениях.



21. Какие проблемы возникают в пространствах очень больших размерностей? Как метод главных компонент (Principal component analysis, PCA) помогает с ними бороться? В каких еще задачах можно использовать этот метод?

Curse of Dimensionality (проклятие размерностей) — это проблема, возникающая при возрастании числа измерений в наборе данных (больше десяти).

- Экспоненциальный рост необходимого количества данных для обучения.
- Чем больше измерений в пространстве, тем меньше значат расстояния между точками, то есть расстояние между точками несёт в себе меньше информации. Это можно наглядно показать на примере того, какую часть единичного гиперкуба занимает единичная гипертсфера при возрастающем числе измерений: это отношение будет стремиться к нулю. То есть всё бóльшая часть объёма находится в «углах» гиперкуба. (На самом деле нихуя не наглядно, но Закиров объяснял так.)

PCA на примере плоскости: хотим двумерные точки урезать в одномерные. Для этого их можно просто спроецировать на ось абсцисс, но тогда потеряется много информации. Однако, если эти точки примерно лежат на какой-нибудь прямой, то можно совершить такое преобразование плоскости, чтобы эти точки оказались примерно на оси абсцисс. Тогда их можно спроецировать на неё без больших потерь информации. Таким образом, PCA переносит как можно больше информации в первые измерения, позволяя просто выкинуть часть последних без особых потерь и тем самым уменьшить количество измерений.

В случае, который показан на рисунке, нужно повернуть плоскость так, чтобы ось абсцисс совпала с прямой AB, а после этого спроецировать на неё точки. В итоге из двумерных точек получим одномерные, сохранив при этом максимум информации.

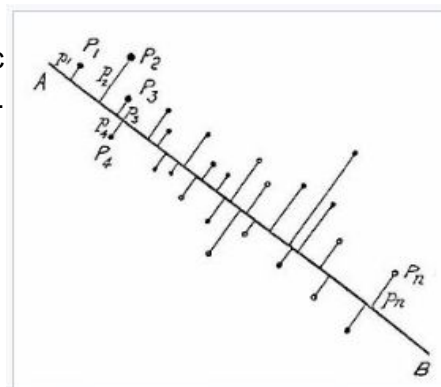


Иллюстрация к знаменитой работе Пирсона (1901): даны точки P_i на плоскости, p_i — расстояние от P_i до прямой AB . Ищется прямая AB , минимизирующая сумму $\sum_i p_i^2$

22. Что такое рекомендательные системы (recommender systems), какие задачи они решают и каким образом работают?

Рекомендательные системы — [программы](#), которые пытаются предсказать, какие объекты ([фильмы](#), [музыка](#), [книги](#), [новости](#), [веб-сайты](#)) будут интересны пользователю, имея определенную информацию о его [профиле](#).

Две основные стратегии создания рекомендательных систем — [фильтрация на основе содержания](#) и [коллаборативная фильтрация](#). При фильтрации на основе содержания создаются профили пользователей и объектов, профили пользователей могут включать демографическую информацию или ответы на определённый набор вопросов, профили объектов могут включать названия жанров, имена актёров, имена исполнителей и другую атрибутивную информацию в зависимости от типа объекта. Например, в Music Genome Project музыкальный аналитик оценивает каждую композицию по сотням различных музыкальных характеристик, которые могут использоваться для выявления музыкальных предпочтений пользователя. При коллаборативной фильтрации используется информация о поведении пользователей в прошлом — например, информация о покупках или оценках. В этом случае не имеет значения, с какими типами объектов ведётся работа, но при этом могут учитываться неявные характеристики, которые сложно было бы учесть при создании профиля. Основная проблема этого типа рекомендательных систем — «холодный старт»: отсутствие данных о недавно появившихся в системе пользователях или объектах. В процессе работы рекомендательные системы собирают данные о пользователях, используя сочетание явных и неявных методов. Примеры явного сбора данных:

- запрос у пользователя оценки объекта по дифференцированной шкале;
- запрос у пользователя ранжировки группы объектов от наилучшего к наихудшему;
- предъявление пользователю двух объектов с вопросом о том, какой из них лучше;
- предложение создать список объектов, любимых пользователем.

Примеры неявного сбора данных:

- наблюдение за тем, что осматривает пользователь в [интернет-магазинах](#) или [базах данных](#) другого типа;
- ведение записей о поведении пользователя онлайн;
- отслеживание содержимого компьютера пользователя.

Рекомендательные системы сравнивают однотипные данные от разных людей и вычисляют список рекомендаций для конкретного пользователя. Некоторые примеры их коммерческого и некоммерческого использования приведены в статье о [коллаборативной фильтрации](#). Для вычисления рекомендаций используется [граф интересов](#)^[3]. Рекомендательные системы — удобная альтернатива поисковым алгоритмам, так как позволяют обнаружить объекты, которые не могут быть найдены последними. Любопытно, что рекомендательные системы часто используют поисковые машины для индексации необычных данных. [Источник](#)

23. Приведите пример регрессионной задачи и опишите полный цикл ее решения.

Задача: предсказать рейтинг энергопотребления здания в Нью-Йорке по имеющимся в открытом доступе данным. Это целое число от 1 до 100, поэтому будем рассматривать задачу как регрессионную. (Теоретически можно решать её как классификационную с 100 классами. Но, наверно, проще округлять всё же.)

Этапы решения:

1. Очистка данных
 - По разным причинам в данных могут оказаться пропуски. Нужно либо заполнить их, либо удалить. Удалять можно как записи из набора данных, так и признаки целиком, если они содержат очень много пропусков.
 - В данных также могут быть выбросы — значения, которые очень сильно отклоняются от нормы. Возможно, они возникли в результате ошибок при сборе данных, а может быть, это корректные значения. Тем не менее, они не соответствуют норме и должны быть удалены из выборки для лучшей обучаемости.
2. Предварительный анализ данных: поиск различных зависимостей в наборе данных. Например, можно посмотреть, какое распределение имеет целевой признак, чтобы выбрать более подходящую модель. На этом этапе могут строиться различные графики для визуализации зависимостей, вычисляться коэффициенты корреляции
3. Выбор и создание признаков
 - На основе имеющихся признаков могут быть созданы новые. Для числовых значений это можно проделать логарифмированием, взятием корня, возведением в степень и т. д. Для категориальных — кодирование для того, чтобы модель могла работать с такими признаками.
 - Может понадобится отбор признаков. Например, если два признака (ни один из которых не является целевым) сильно коррелируют, то один из них лучше отбросить.
4. Выбор методики оценивания ошибки, например, MSE
5. Обучение модели и проверка результатов
 - В данном случае в качестве модели берётся нейронная сеть. Но вообще можно брать разные модели и выбирать из них ту, которая показывает наилучшие результаты

<https://neurohive.io/ru/tutorial/primer-reshenija-realnoj-zadachi-po-mashinnomu-obucheniju-na-python/>

24. Приведите пример классификационной задачи и опишите полный цикл ее решения.

Задача: предсказать, кто из пассажиров Титаника выживет, а кто нет. На выходе имеем два класса: выжившие и невыжившие, — поэтому задача классификационная.

Этапы решения: эмм... Ctrl+C, Ctrl+V из предыдущего вопроса?

Ещё примеры задач:

http://www.machinelearning.ru/wiki/index.php?title=Классификация#.D0.9F.D1.80.D0.B8.D0.BC.D0.B5.D1.80.D1.8B_.D0.BF.D1.80.D0.B8.D0.BA.D0.BB.D0.B0.D0.B4.D0.BD.D1.8B.D1.85_.D0.B7.D0.B0.D0.B4.D0.B0.D1.87