Порождающие модели в машинном обучении

Д.Е. Намиот, Е.А. Ильюшин

***Аннотация*— В этой статье, написанной для учебной программы по робастному машинному обучению, обсуждаются так называемые порождающий (генеративные) модели в машинном обучении. Генеративные модели изучают распределение данных из некоторого выборочного набора данных, а затем могут генерировать (создавать) новые экземпляры данных. Генеративные модели являются популярными инструментами с широким спектром приложений. Недавние достижения в области глубокого обучения привели к улучшениям в архитектуре генеративных моделей, и некоторые современные модели могут (в некоторых случаях) давать достаточно реалистичные результаты, чтобы обмануть как конечных пользователей (людей), так и алгоритмы распознавания и классификации. Генеративные модели используются при конструировании состязательных атак. Вместо того, чтобы искать минимальные модификации, как в классических атаках уклонением, порождающие модели позволяют, например, создавать состязательные примеры полностью с нуля. В то же самое время, порождающие модели так же уязвимы для состязательных атак, как и классификаторы. Это наш первый материал по данной теме и, очевидно, рассмотрение этой важной темы будет продолжено.**

***Ключевые слова*—генеративные модели, порождающие модели, машинное обучение.**

I. Введение

Порождающие модели (generative models) – класс статистических моделей, основанных на анализе распределения самих данных. Для набора данных X и набора меток Y, генеративные модели фиксируют совместную вероятность p (X, Y) или просто p (X), если меток нет. То есть, они оценивают вероятность появления собственно данных. Это отличает их от дискриминантных моделей, которые отражают условную вероятность p(X | Y) – то есть, вероятность оценки (классификации) данных. Классификаторы, которые порождают решения без вычисления вероятностей, также относятся к дискриминантным моделям [1].

Порождающие модели вычисляют распределение вероятностей с помощью обучающих примеров. Соответственно, исходя из слепков (отсчетов) воссозданного многомерного распределения, порождающие модели создают новые наборы данных, подобные обучающим примерам. Отсюда и название – порождающие. Например, обучаем порождающую модель на реальных изображениях лиц и синтезируем новые изображения похожих лиц.

Кратко: генеративные модели могут генерировать новые экземпляры данных, а дискриминантные модели различают разные типы экземпляров данных.

Генеративная модель включает в себя распределение самих данных и показывает, насколько вероятен данный пример. Например, модели, которые предсказывают следующее слово в последовательности, обычно являются генеративными моделями, потому что они могут назначать вероятность последовательности слов.

Дискриминантная модель игнорирует вопрос о том, вероятен ли данный экземпляр, и просто сообщает вам, насколько вероятно, что метка применима (относится) к этому экземпляру.

Есть понятие Deep Generative models (DGM) - глубокие генеративные модели [2, 3]. DGM формируются за счет комбинации генеративных моделей и глубоких нейронных сетей. Идея DGM заключается в том, что нейронные сети, используемые в качестве генеративных моделей, имеют набор параметров, значительно меньших, чем объем данных, используемых для их обучения, поэтому модели вынуждены обнаруживать и эффективно усваивать сущности данных.

Генеративные модели сложны. Генеративные модели решают более сложную задачу, чем аналогичные дискриминантные модели. Генеративные модели должны моделировать больше. Генеративная модель изображений может улавливать корреляции типа «вещи, похожие на лодки, вероятно, будут появляться рядом с вещами, которые выглядят как вода» и т.д. Это очень сложные дистрибутивы.

Напротив, дискриминантная модель может узнать разницу между «легковой автомобиль» и «грузовой автомобиль», просто отыскав несколько характерных закономерностей. Дискриминантная модель может игнорировать многие корреляции, которые генеративная модель должна правильно устанавливать.

Дискриминантные модели пытаются провести границы в пространстве данных, в то время как генеративные модели пытаются смоделировать, как данные размещаются в пространстве. Например, на следующей картинке (источник – Google) показаны дискриминантные и порождающие модели рукописных

Статья получена 21 мая 2022.
Д.Е. Намиот - МГУ имени М.В. Ломоносова (e-mail: dnamiot@gmail.com).
Е.А. Ильюшин - МГУ имени М.В. Ломоносова (email: john.ilyushin@gmail.com)

цифр:

Два рисунка, один помечен как «Дискриминантная модель», а другой - «Генеративная модель». Оба графика показывают те же четыре точки данных. Каждая точка помечена изображением рукописной цифры, которую он представляет. В дискриминантной модели есть пунктирная линия, отделяющая две точки данных от двух других. Область над пунктирной линией помечена как «y = 0» и область под линией помечена как «y = 1». В генеративной модели два пунктирных круга нарисованы вокруг двух пар точек. При этом верхний кружок помечен как «y = 0», а нижний круг помечен как «y = 1».

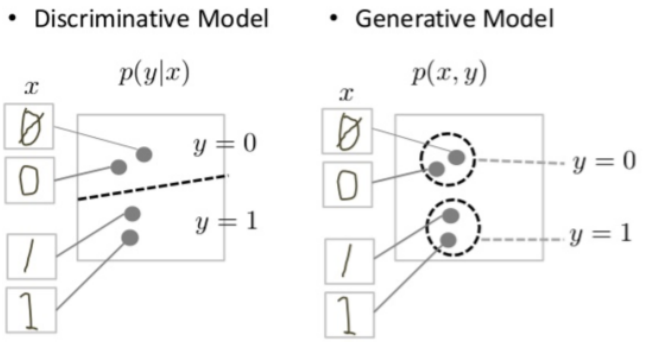

Рис. 1. Дискриминантная и генеративная модели распознавания рукописных цифр [1].

Дискриминантная модель пытается определить разницу между написанными от руки 0 и 1, рисуя линию в пространстве данных. Если модель правильно построит разделитель, она может отличить 0 от 1 без необходимости точно моделировать, где экземпляры размещаются в пространстве данных по обе стороны от разделителя. Напротив, генеративная модель пытается произвести убедительные (похожие на настоящие) единицы и нули, генерируя цифры, которые близки к их реальным аналогам в пространстве данных. Генеративная модель должна моделировать распределение в пространстве данных.

В отличие от дискриминантных моделей, используемых для задач классификации или регрессии, порождающие модели вычисляют распределение вероятностей с помощью обучающих примеров. Извлекая отсчеты этого многомерного распределения, порождающие модели создают новые образцы, подобные обучающим. Это означает, что порождающая модель, обученная на реальных изображениях лиц, может синтезировать новые изображения похожих лиц.

Подробнее об устройстве таких моделей рассказал Йен Гудфеллоу (Ian Goodfellow) в 2016 году в своем обучающем курсе по NIPS [4]. Описанная им архитектура, порождающая состязательная сеть (GAN), особенно популярна сейчас в научном мире, поскольку позволяет приблизиться к неконтролируемому обучению. Сети GAN состоят из двух нейросетей: генератора, который получает на вход случайных шум и должен синтезировать контент (например, изображения), и дискриминатора, который обучен с помощью изображений и должен отделять реальные изображения от ненастоящих, созданных генератором.

Состязательные сети можно понимать как игру, по правилам которой генератор должен постепенно научиться создавать из шума такие изображения, которые дискриминатор не сможет отличить от реальных. Постепенно этот подход стал применяться для самых разных типов данных и задач [1].

GAN - это всего лишь один из видов генеративной модели. Кроме GAN есть, конечно, и другие типы и примеры генеративных моделей.

II. GAUSSIAN MIXTURE MODEL

В этом разделе рассматривается Gaussian mixture model (GMM) [5] и другие типы смешанных моделей.

В статистике, смешанная модель представляет собой вероятностную модель для представления присутствия подгрупп (суб-популяций) в общей популяции, которая не требует, чтобы наблюдаемый набор данных определял суб-популяцию, к которой принадлежит данное отдельное наблюдение. Или, другими словами, модель относит каждое наблюдение к некоторому кластеру путем максимизации апостериорной вероятности, что точка данных принадлежит своему присвоенному кластеру.

Формально, модель смеси соответствует распределению смеси, которое представляет собой распределение вероятностей наблюдений в генеральной совокупности. Однако, в то время как проблемы, связанные со «смешанными распределениями», связаны с получением свойств общей популяции на основе характеристик подгрупп, «смешанные модели» используются для статистических выводов о свойствах подгрупп с учетом только наблюдений за объединенными данными, без информации об идентичности подгруппы данных.

Типичная модель смеси представляет собой иерархическую модель, состоящую из следующих компонентов [6]:

• Наблюдаемые N случайных величин, каждая из которых распределена в соответствии со смесью K компонентов, причем компоненты принадлежат к одному и тому же параметрическому семейству распределений (например, все нормальные), но с разными параметрами

• N случайных скрытых переменных, определяющих идентичность компонента смеси каждого наблюдения, каждая из которых распределена согласно K -мерному категориальному распределению (дискретному распределению вероятностей, которое описывает возможные результаты случайной величины, принимающей одно из K возможных значений - обобщенное распределение Бернулли)

• Набор из K весов смеси, которые представляют собой вероятности, сумма которых равна 1.

• Набор из K параметров, каждый из которых определяет параметр соответствующего компонента смеси. Во многих случаях каждый «параметр» на самом деле представляет собой набор параметров. Например, если компоненты смеси являются гауссовскими распределениями, для каждого компонента будет среднее значение и дисперсия. Если компоненты смеси являются категориальными распределениями (например, когда каждое наблюдение является маркером из конечного алфавита размера V), будет вектор вероятностей V, компоненты которого дают в сумме 1.

Почему именно Гаусс? В силу центральной предельной теоремы: сумма достаточно большого количества слабо зависимых случайных величин, имеющих примерно одинаковые масштабы, имеет распределение, близкое к нормальному. Так как многие случайные величины в приложениях формируются под влиянием нескольких слабо зависимых случайных факторов, их распределение считают нормальным. При этом должно соблюдаться условие, что ни один из факторов не является доминирующим (то есть ни одно из слагаемых не доминирует и не вносит в сумму определяющего вклада).

Кластеризация с использованием GMM может быть проиллюстрирована довольно понятным способом [7, 8].

Допустим, у нас есть три смеси (K=3). Для произвольной точки (отмечена красным), GMM вычисляет вероятность попадания в один из кластеров

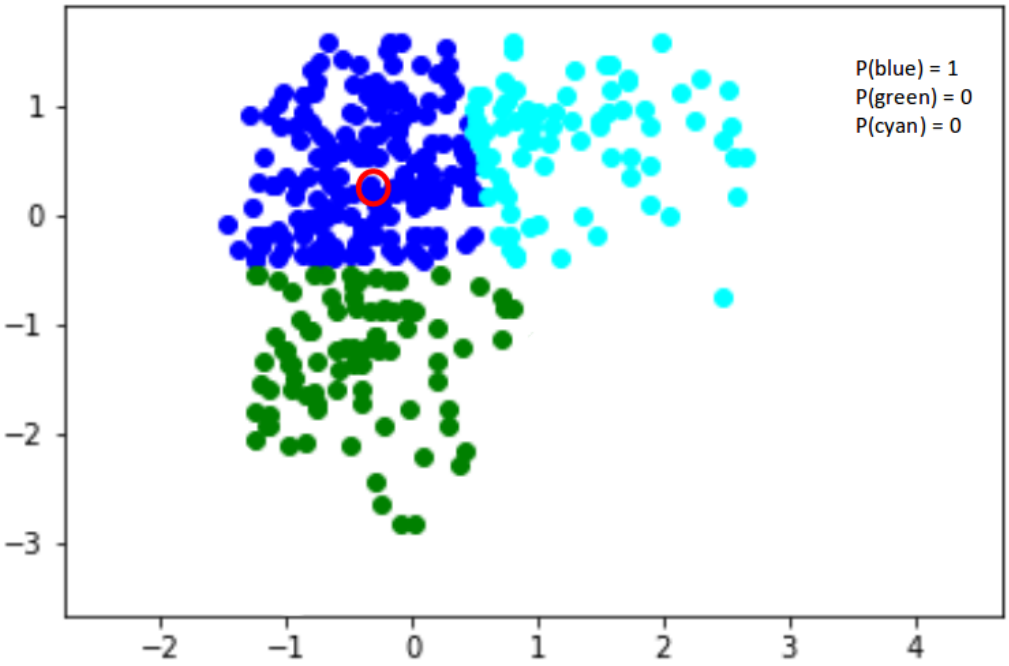

Рис.2. GMM кластеризация [8].

Количество кластеров – это количество нормальных распределений в нашей смеси. Каждое распределение имеет некоторое среднее и дисперсию. Вычисление было бы простым, если бы кластеризация была бы известна. Но нам известно только количество кластеров (мы изначально решили, на сколько групп мы хотим разбить наши данные), но неизвестны параметры этих групп (параметры распределений для каждой из групп). Эти значения определяются с помощью метода, называемого максимизацией ожиданий (EM - Expectation-Maximization). EM - это статистический алгоритм для поиска правильных параметров модели. Обычно EM используется тогда, когда в данных есть пропущенные значения (другими словами - когда данные неполны). Эти пропущенные (недостающие) переменные называются скрытыми переменными.

Поскольку у нас нет значений для скрытых переменных, максимизация ожидания пытается использовать существующие данные для определения оптимальных значений для этих переменных, а затем находит параметры модели. На основе этих параметров модели мы возвращаемся назад и обновляем значения скрытой переменной, после чего процесс повторяется. Цель – максимизация логарифмической функции правдоподобия.

В целом алгоритм максимизации ожиданий состоит из двух шагов:

E-шаг: на этом этапе доступные данные используются для оценки (фактически – предположения, оценивания) значений отсутствующих переменных.

M-шаг: на основе оценочных значений, сгенерированных на E-шаге, полные данные используются для обновления параметров.

Ожидание-максимизация является основой многих алгоритмов, в том числе гауссовских смешанных моделей.

Почему это относят к генеративным моделям? Потому что, оценив распределение входных данных, мы можем создавать (порождать) новые данные, в соответствии с найденным распределением. Примеры соответствующего кода на Python есть, например, в

работе [9].

III Hidden Markov model

В этом разделе рассматривается Hidden Markov model (HMM) [10] - Скрытая марковская модель (СММ). Это статистическая модель, имитирующая работу процесса, похожего на марковский процесс с неизвестными параметрами, в которой задачей ставится определение неизвестных параметров на основе наблюдаемых. Полученные параметры могут быть использованы в дальнейшем анализе, например, для распознавания образов. СММ может быть рассмотрена как простейшая байесовская сеть доверия.

В обычной марковской модели состояние видимо наблюдателю, поэтому вероятности переходов — единственный параметр.

Генерация в Марковских моделях – это генерация последовательностей, в соответствии с вероятностями событий:

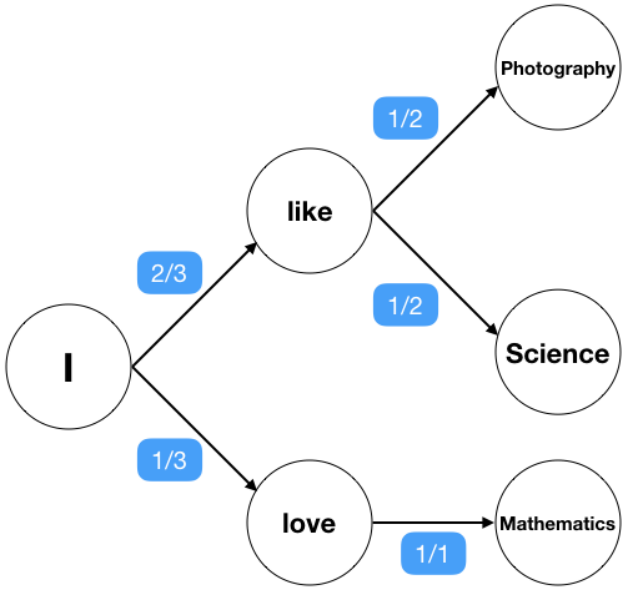

Рис. 3. HMM [11]

В скрытой марковской модели мы можем следить лишь за переменными, на которые оказывает влияние данное состояние.

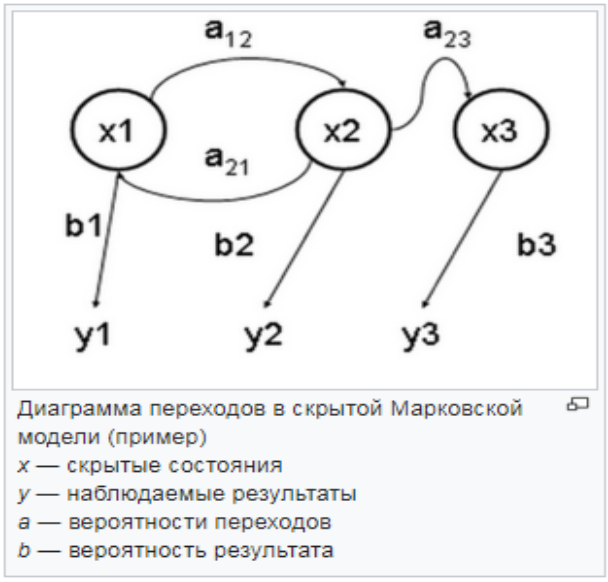

Диаграмма переходов в скрытой Марковской модели (пример)
x — скрытые состояния
y — наблюдаемые результаты
a — вероятности переходов
b — вероятность результата

Рис. 4. Диаграмма переходов в СММ [12]

Каждое состояние имеет вероятностное распределение среди всех возможных выходных значений. Поэтому последовательность символов, сгенерированная СММ, даёт информацию о последовательности состояний.

Формально, скрытая марковская модель может быть определена следующим образом.

Пусть Y $\in$ $\{y_1, y_2. y_3, \ldots y_n\}$ - набор переменных-результатов

$<Y_1, Y_2, \ldots Y_{t-1}, Y_t>$ - их последовательность

Z $\in$ $\{z_1, z_2. z_3, \ldots y_k\}$ – скрытые состояния

СММ определяется как $\theta = (\pi, A, B)$ где

π - набор начальных состояний,

$A = P(Z_{t-1} \mid Z_t)$ - вероятности переходов

$B = P(Y_t \mid Z_t)$ - вероятности результатов

В литературе отмечаются три основные модели применения СММ

1) Оценка правдоподобия: имея последовательность наблюдений $\langle Y_1, \ldots, Y_{t-1}, Y_t \rangle$ и зная модель θ определить $P(Y_1, \ldots Y_t \mid \theta)$.

2) Проблема декодирования: имея последовательность наблюдений и модель θ определить наиболее вероятную последовательность скрытых состояний

$$\arg\max_{Z_1,\ldots Z_t} P(Z_1, \ldots, Z_t, Y_1, \ldots, Y_t \mid \theta)$$

3) Проблема обучения: имея последовательность наблюдений и набор скрытых состояний определить параметры сети, использую оценку максимального правдоподобия

Вот пункт 2 как раз и определяет генеративные возможности СММ. Это генеративная модель, поскольку в ней мы пытаемся воссоздать исходный процесс генерации, ответственный за создание пар скрытое состояние – наблюдаемый результат. В качестве

генеративной модели CMM пытаются смоделировать наиболее вероятную последовательность скрытых состояний с учетом последовательности наблюдений, максимизируя совместную вероятность наблюдений и скрытых состояний. Примеры работы с пошаговыми комментариями могут быть найдены в работе [13].

IV Probabilistic context-free grammar

Probabilistic context-free grammar (PSFG) [14] (вероятностная контекстно-свободная грамматика) – это контекстно-свободная грамматика (частный случай формальной грамматики, у которой левые части всех продукций являются одиночными нетерминалами - объектами, обозначающими какую-либо сущность языка), в которой каждому правилу вывода соответствует вероятность.

Вероятность вывода определяется как произведение вероятностей используемых в нём правил вывода, таким образом, некоторые выводы лучше соответствуют стохастической грамматике, чем другие. СКС-грамматики расширяют КС-грамматики так же, как скрытые марковские модели расширяют регулярные грамматики.

Основные идеи использования вероятностей в грамматике:

1. Устранение синтаксической неоднозначности. Неизбежна двусмысленность в естественных языках. Вероятность – отражение этой двусмысленности.
2. Покрытие (новая трактовка). Появилась новая интерпретация, которую мы хотим использовать, пусть и с небольшой вероятностью.
3. Репрезентативность. Предположим, мы хотим адаптировать синтаксический анализатор (грамматику) к новому домену (новой предметной области), в котором некоторые слова используются по-разному. Соответственно, нужно как-то различать варианты использования.

Формально, такая грамматика может быть определена как набор G = (M, T, R, P, S), где

M – множество нетерминальных символов
T – множество терминальных символов
R – набор продукций (правил)
S – начальный символ
P набор вероятностей для продукций

Схема порождения очевидна – строим вероятные последовательности.

Пример использования из работы [15] - установление авторства текстового документа, используя вероятностные контекстно-свободные грамматики. Этот подход предполагает построение вероятностной контекстно-свободной грамматики для каждого автора. Далее эта грамматика используется как языковая модель для классификации. Далее для произвольного текстового документа определяется наиболее вероятный автор, грамматике которого соответствует данный документ. PSFG грамматика строится посредством так называемого статистического парсера. Это парсер, который из имеющихся предложений строит все возможные грамматические деревья, снабжая их вероятностями.

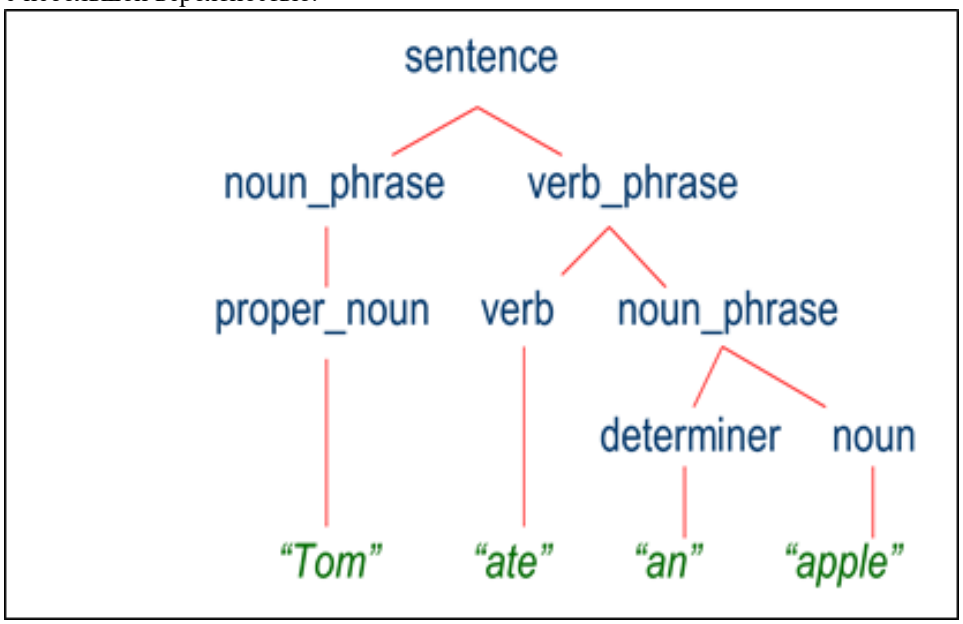

Рис. 5. PSFG [15]

В качестве практического инструмента можно назвать Apache OpenNLP (https://opennlp.apache.org/).

V Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) [16] – это порождающая модель, позволяющая объяснять результаты наблюдений с помощью некоторых неявных групп (тем, топиков). Например, если наблюдениями

являются слова, собранные в документы, то утверждается, что каждый документ представляет собой смесь небольшого количества тем и появление каждого слова связано с одной из тем документа. Благодаря этому возможно выявление причин сходства некоторых документов. LDA является одним из методов тематического моделирования.

В LDA каждый документ может рассматриваться как набор различных тематик. В LDA предполагается, что распределение тематик имеет в качестве априори распределения Дирихле.

Обучение – проводим тематический анализ и определяем набор тем.

Генерация: мы создаем документ, подобный имеющемуся (документ с тем же набором тем)

Основная идея LDA, как указано выше, состоит в том, что документы представляют собой случайные комбинации скрытых тем, каждая из которых характеризуется распределением по словам. На рисунке 6 представлен процесс генерации документа LDA, где узлы представляют случайные величины.

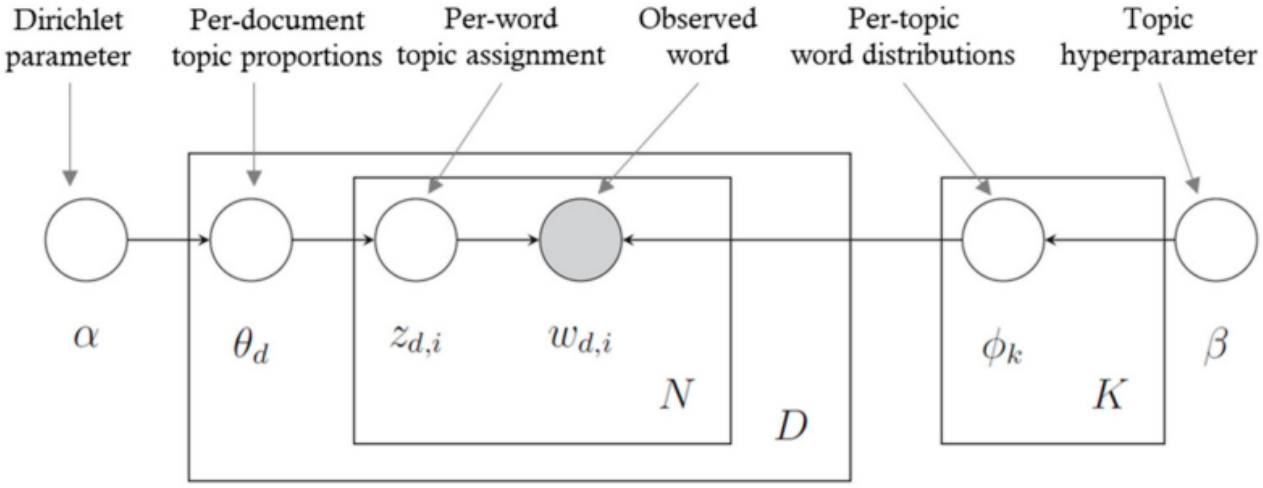

Рис. 6. Генерация документа [16]

Заштрихованный узел w_d, обозначающий слово в d-м документе, только которое является наблюдаемым, тогда как другие узлы скрыты. Процесс создания документов LDA выглядит следующим образом: (1) распределение слов по темам (φ_k) оценивается для всего корпуса (датасета, на котором проводилось обучение); (2) определяются пропорции темы по каждому документу (θ_d); (3) каждое слово выбирается на основе заданной темы ($z_{d,i}$). На рисунке 1 α и β - гиперпараметры Дирихле для θ_d и φ_k соответственно. N обозначает количество слов в документе, D обозначает количество документов в корпусе, K - количество тем в корпусе, соответственно

VI Bayesian network

Bayesian network (e.g. Naive bayes, Autoregressive model) [18] - графовая вероятностная модель, представляющая собой множество переменных и их вероятностных зависимостей по Байесу. Например, байесовская сеть может быть использована для вычисления вероятности того, чем болен пациент, по наличию или отсутствию ряда симптомов, основываясь на данных о зависимости между симптомами и болезнями.

Формально, байесовская сеть — это ориентированный ациклический граф, каждой вершине которого соответствует случайная переменная, а дуги графа кодируют отношения условной независимости между этими переменными. Вершины могут представлять переменные любых типов, быть взвешенными параметрами, скрытыми переменными или гипотезами.

Если переменные байесовской сети являются дискретными случайными величинами, то такая сеть называется дискретной байесовской сетью. Байесовские сети, которые моделируют последовательности переменных, называют динамическими байесовскими сетями. Байесовские сети, в которых могут присутствовать как дискретные переменные, так и непрерывные, называются гибридными байесовскими сетями. Байесовская сеть, в которой дуги помимо отношений условной независимости кодируют также отношения причинности, называют причинно-следственными байесовыми сетями (causal bayesian networks) [20].

Генерация новых данных по существующему графу начинается с некоторого корневого узла. По его связям порождаются новые узлы. Затем следуют дочерние элементы в зависимости от их родителей до тех пор, пока не будут получены данные для всех узлов.

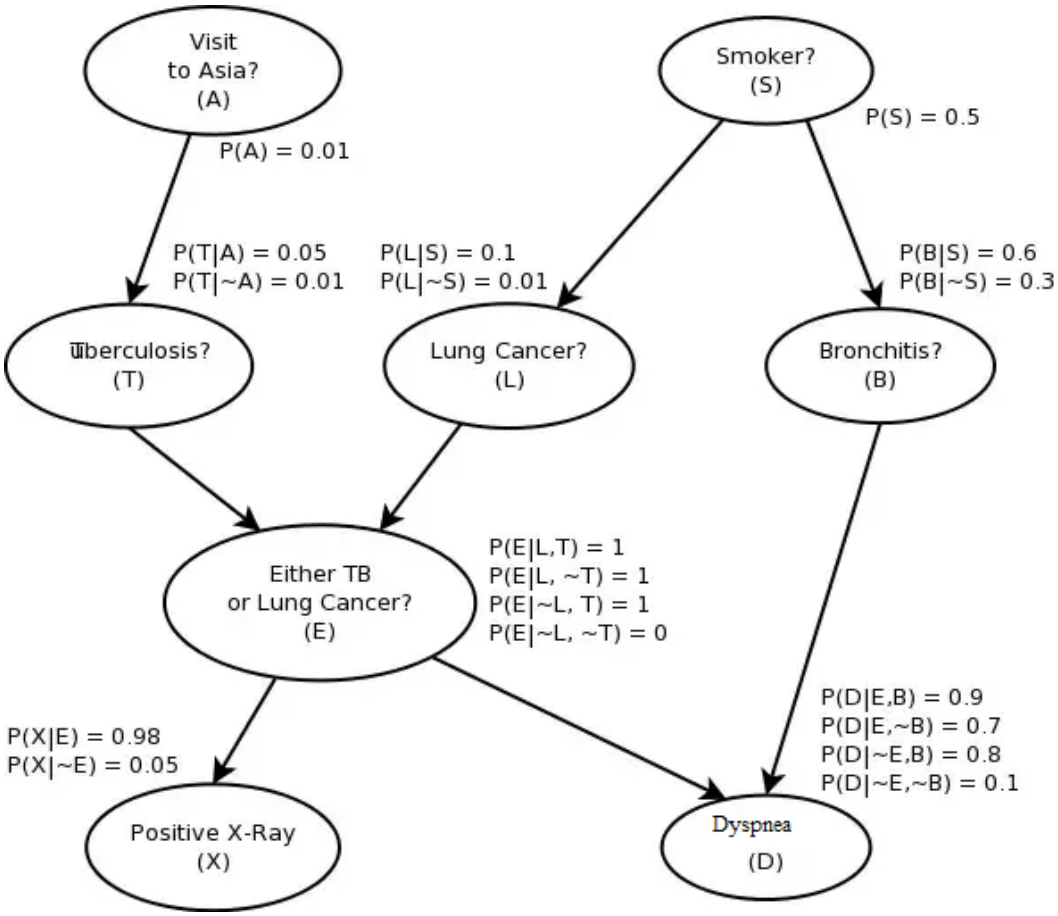

Рис.7. Байесовская сеть [19]

VII Boltzmann machine

Boltzmann machine (e.g. Restricted Boltzmann machine, Deep belief network) [21] - вид стохастической рекуррентной нейронной сети. В статистике - это случайные марковские поля. Эта сеть использует для обучения алгоритм имитации отжига (Simulated annealing) - общий алгоритмический метод решения задачи глобальной оптимизации. Это один из примеров методов Монте-Карло. Алгоритм основывается на имитации физического процесса, который происходит при кристаллизации вещества, в том числе при отжиге металлов. Предполагается, что атомы уже выстроились в кристаллическую решётку, но ещё допустимы переходы отдельных атомов из одной ячейки в другую. Предполагается, что процесс протекает при постепенно понижающейся температуре. Переход атома из одной ячейки в другую происходит с некоторой вероятностью, причём вероятность уменьшается с понижением температуры. Устойчивая кристаллическая решётка соответствует минимуму энергии атомов, поэтому атом либо переходит в состояние с меньшим уровнем энергии, либо остаётся на месте. На каждом шаге алгоритм вычисляет новую точку и понижает значение величины (изначально положительной), понимаемой как «температура». Алгоритм останавливается по достижении точки, которая оказывается при температуре ноль.

Описанная схема похожа на градиентный спуск, но будет работать при множестве локальных экстремумов. Машина Больцмана представлена на рисунке 8.

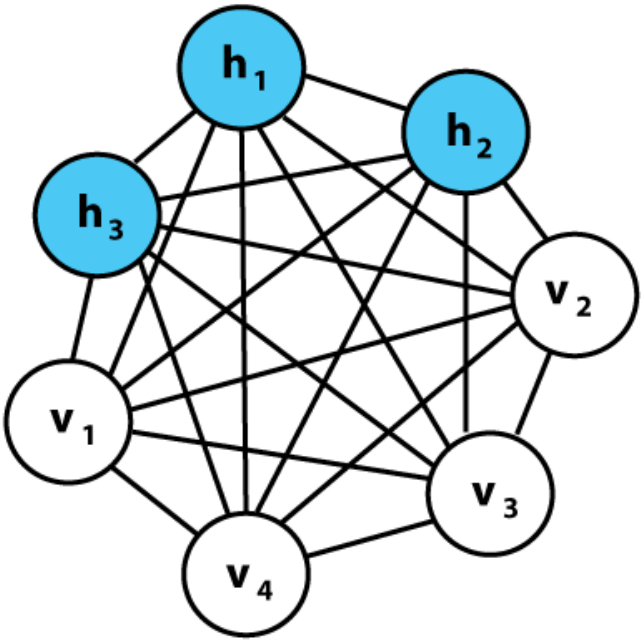

Рис. 8. Машина Больцмана [22]

Это сеть с симметричными связями и отсутствием связей узла с самим собой. В машине Больцмана два

типа узлов: видимые (v_i на рисунке) и скрытые (h_i на рисунке).

Ограниченная машина Больцмана – это двудольный граф, где видимые узлы соединены только с невидимыми, но не между собой

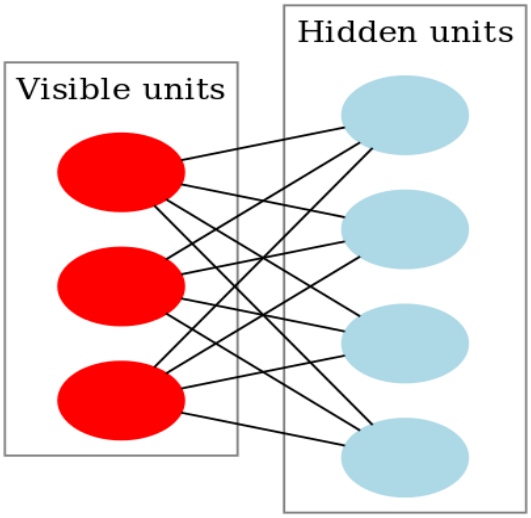

Рис. 9. Ограниченная машина Больцмана [22]

Использование ограниченной машины Больцмана - это алгоритмы обучения без учителя, которые пытаются реконструировать (то есть – сгенерить) вводимые пользователем данные. Для этого они пытаются изучить закономерности на примерах из исходных данных (данных для обучения). Затем это используется для создания некоторого шаблона, который может быть использован для восстановления приближений исходного ввода.

Пример – рекомендательная система по подбору книг. Определяется (порождается) шаблон для определения читательского набора предпочтений, который далее используется для выработки рекомендаций.

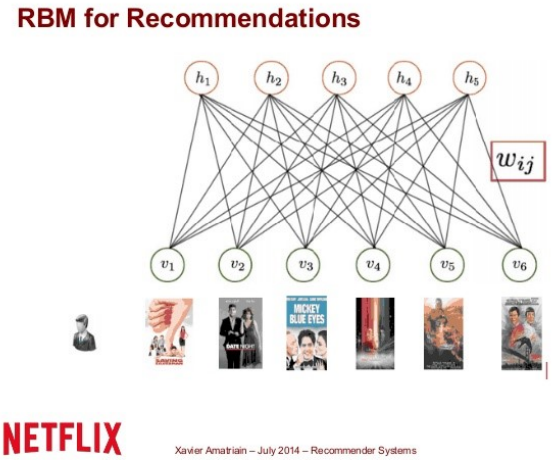

Рис. 10. Ограниченная машина Больцмана в рекомендательных системах [23].

VIII Variational autoencoder

Классические автокодировщики [24] - это простые нейронные сети с прямой связью, состоящие из трех частей: кодировщика, скрытого слоя (бутылочное горло) и слоев декодера. Кодировщик - это стандартная искусственная нейронная сеть, которая принимает изображение на входе и выводит вектор признаков, который в случае автокодировщика называется бутылочным горлом или скрытым представлением. Здесь используется термин скрытый, поскольку этот вектор теперь содержит всю значимую информацию, необходимую для восстановления исходного изображения (X). Задача декодера - принять этот скрытый вектор и восстановить исходное изображение ($X`$). Затем сеть обучается на основе оценки потери реконструкции между $X`$ (созданное изображение) и X (исходное изображение).

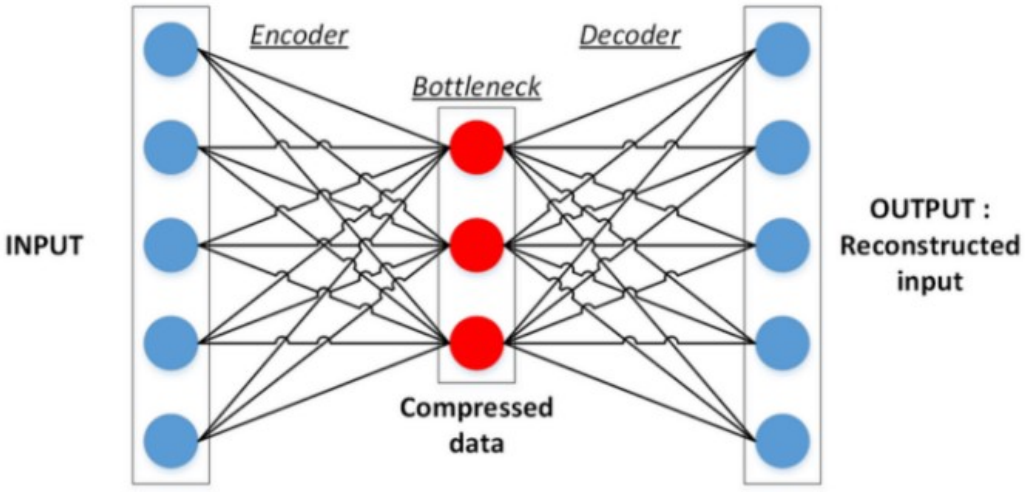

Рис. 11. Variational AutoEncoders (VAE) [25]

Отметим, что размеры входного и выходного слоя – одинаковые, а скрытый слой – меньше их.

Скрытый слой, таким образом, понижает размерность, распознавая наиболее важные параметры.

Математически, пусть g(·) представляет собой кодировщик, параметризованный с помощью ϕ, и f(·) представляет собой декодер, параметризованный с помощью θ. Скрытый вектор z тогда представляется как $g_\phi(X)$. Декодер получает вход z и выводит реконструированный вход, который обозначим как $X`$. $X`$ может быть представлен как $f_\theta(g_\phi(X))$. Тогда функция потерь есть $L(\theta, \phi, X, X`)$, что в случае MSE выглядит так:

$$L_{AE}(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_\theta(g_\phi(x^{(i)})))^2$$

Основная идея автокодировщика состоит в том, что, в отличие от метода главных компонент (также понижающем размерность), он способен изучать нелинейные многообразия.

Чаще всего автокодировщики применяют каскадно для обучения глубоких (многослойных) сетей без учителя. Для этого слои обучаются друг за другом, начиная с первых. К каждому новому необученному слою на время обучения подключается дополнительный выходной слой, дополняющий сеть до архитектуры автокодировщика, после чего на вход сети подается набор данных для обучения. Веса необученного слоя и дополнительного слоя автокодировщика обучаются при помощи метода обратного распространения ошибки. Затем слой автокодировщика отключается и создается новый, соответствующий следующему необученному слою сети. На вход сети снова подается тот же набор данных, обученные первые слои сети остаются без изменений и работают в качестве входных для очередного обучаемого слоя автокодировщика.

Модель относится к порождающим, поскольку скрытый слой, например, для изображений создает новое (сжатое) изображение. Но – изображения будут одинаковыми. Какой-либо вход будет всегда создавать одну и ту же скрытую переменную и, соответственно, всегда один и тот же выход. Вариационный автокодировщик добавляет случайные значения на этапе кодирования, так, что если мы несколько раз повторяем ввод, мы будем получать разные значения за счет случайности.

Вариационный кодировщик выдаёт не один вектор размера n (n – размер скрытого слоя), а два вектора размера n – вектор средних значений μ и вектор стандартных отклонений σ.

Вариационные автокодировщики формируют параметры вектора длины n из случайных величин Xi, причем i-е элементы векторов μ и σ являются средним и стандартным отклонением i-й случайной величины Xi. Вместе эти величины образуют n-мерный случайный вектор, который посылается на декодер для восстановления данных. Это и означает, что даже для одинаковых входных данных результат кодирования будет разным вследствие случайности выбора вектора кодирования.

Среднее значение вектора определяет точку, вблизи которой будет вершина вектора, в то время как стандартное отклонение определяет, насколько далеко может отстоять вершина от этого среднего. Таким образом, вершина вектора кодирования может лежать внутри n-мерного круга. Поэтому входному объекту соответствует уже не одна точка в скрытом пространстве (как для стандартного кодировщика), а некоторая непрерывная область.

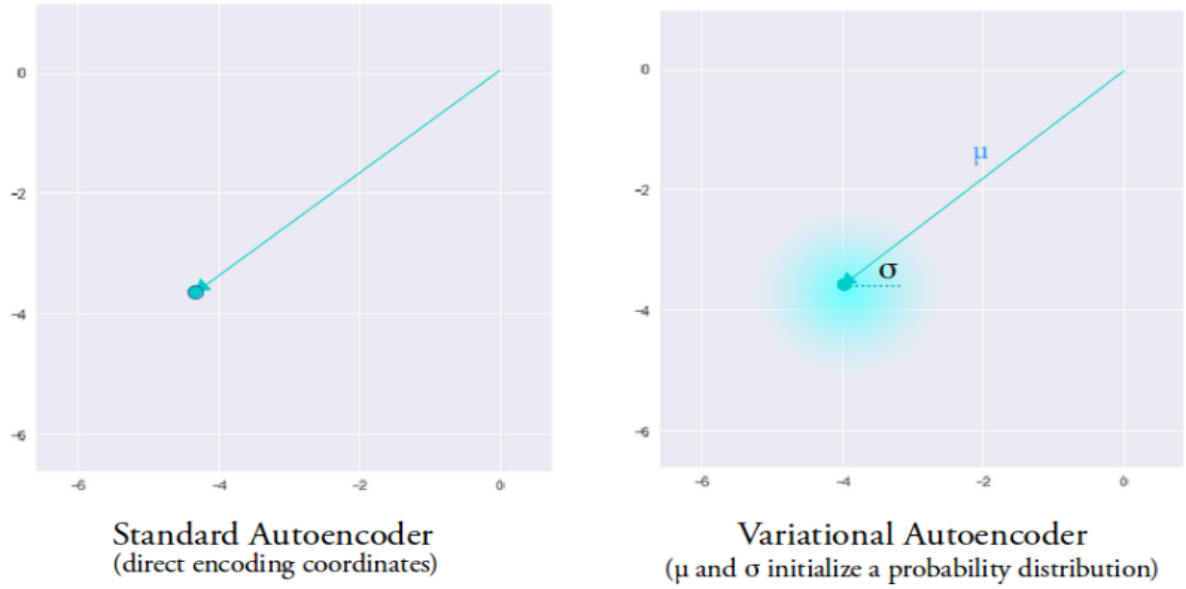

Рис. 12. Стандартный и скрытый автокодировщики [26]

Этот факт позволяет декодеру работать не с одним единственным вектором кодирования, соответствующим входным данным, а с их набором, благодаря чему в восстановление даже одного изображения вносится доля вариативности. Теперь модель обладает вариативностью даже в пределах одного вектора кодирования, так как скрытое пространство локально непрерывно, т.е. непрерывно для каждого образца входных данных.

Традиционный автокодировщик:

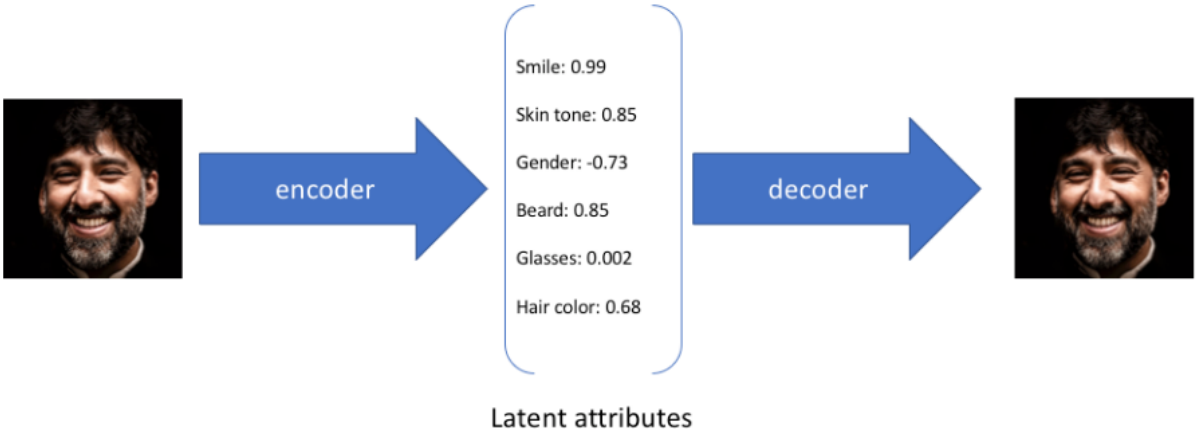

Рис. 13. Стандартный автокодировщик [26]

И вариационный автокодировщик

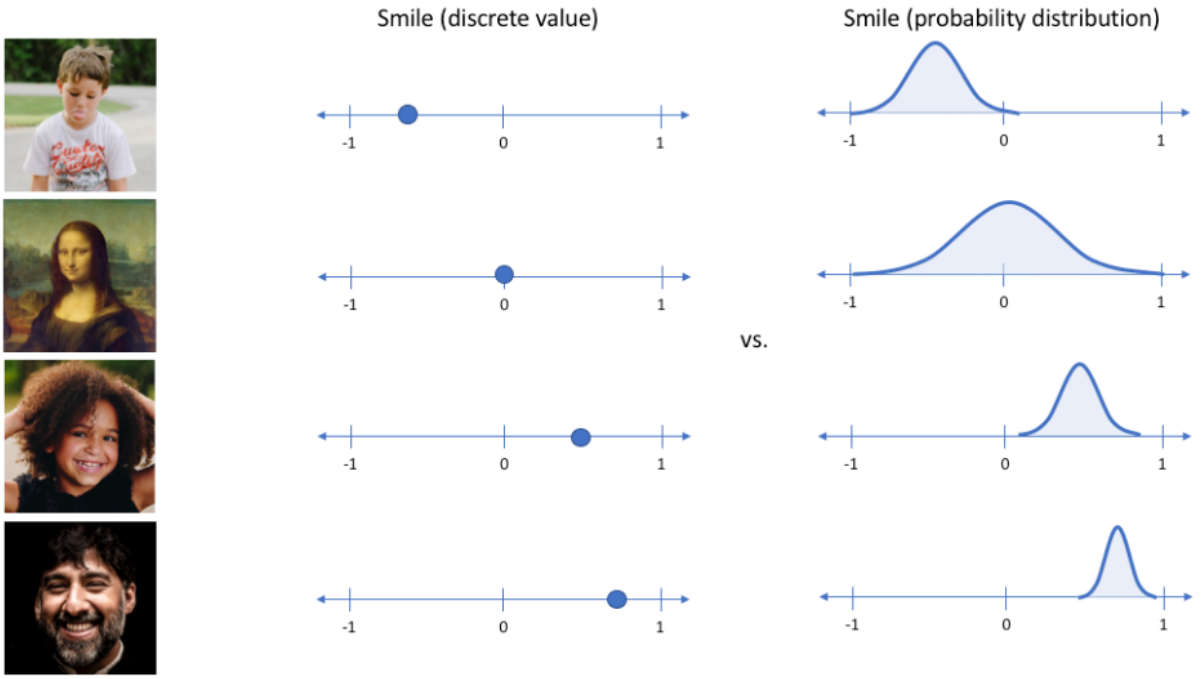

Рис. 14. Вариационный автокодировщик [26]

Каждое значение берется из некоторого распределения (нормального распределения).

Создавая нашу модель кодировщика для вывода диапазона возможных значений (статистического распределения), из которых мы будем случайным образом производить выборку для подачи в нашу модель декодера, мы, по сути, обеспечиваем непрерывное, гладкое представление скрытого пространства. Мы ожидаем, что для любой выборки скрытых распределений наша модель декодера сможет точно реконструировать входные данные. Таким образом, значения, которые находятся рядом друг с другом в скрытом пространстве, должны

соответствовать очень похожим реконструкциям.

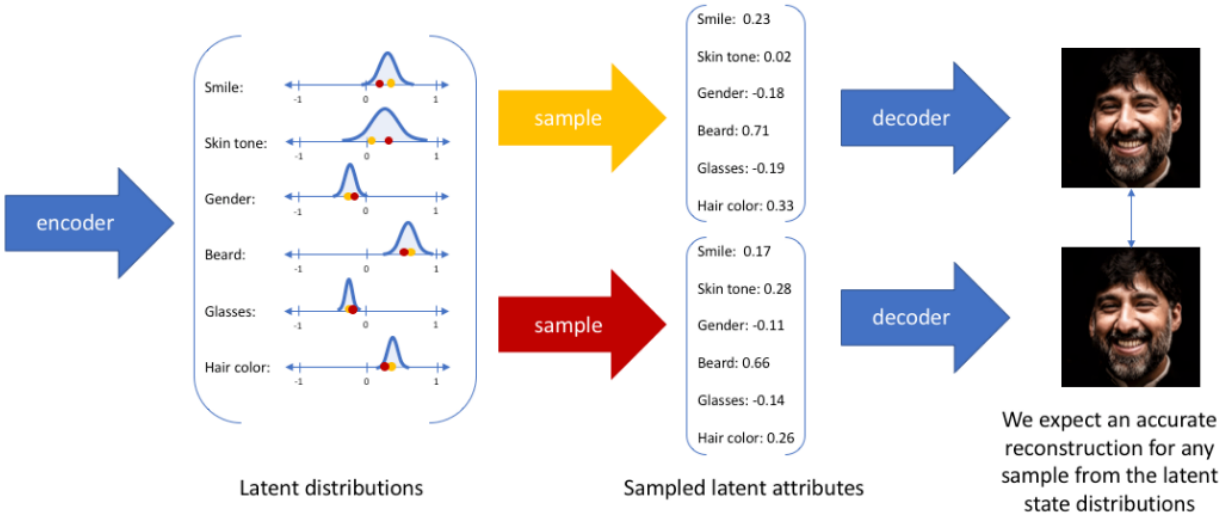

Рис. 15. Декодирование скрытого слоя [26]

Предположим, что у нас есть скрытая переменная z, которая определяет некоторое наблюдаемое значение x. Мы можем видеть только x, но мы хотели бы вывести характеристики z. Другими словами, мы хотим вычислить $p\ (z \mid x)$.

$$p\ (z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

тогда

$$p\ (x) = \int p(x \mid z)p(z)dz$$

Обычно это оказывается трудноразрешимым распределением. Однако мы можем применить вариационный вывод для оценки этого значения.

Дивергенция Кульбака-Лейблера - мера схожести между двумя распределениями. Используется для приближения «сложного» распределения $p(x)$ более «простым» распределением $q(x)$.

$$KL(p \,||\, q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

$$KL(p \,||\, q) \geq 0$$

$$KL(p \,||\, q) = 0 \Leftrightarrow p = q$$

$$KL(p \,||\, q) \neq KL(q \,||\, p)$$

Давайте аппроксимируем $p(z|x)$ другим распределением $q(z|x)$, которое мы определим так, чтобы оно имело управляемое распределение. Если мы можем определить параметры $q\ (z|x)$ так, чтобы они были очень похожи на $p\ (z|x)$, мы могли бы использовать их для выполнения приближенного вывода о трудноразрешимом распределении.

Дивергенция KL - это мера разницы между двумя распределениями вероятностей. Таким образом, если бы мы хотели убедиться, что $q\ (z \mid x)$ было похоже на p (z | x), мы могли бы минимизировать KL-расхождение между двумя распределениями.

$$\min KL(q(z|x)||p(z|x))$$

Это эквивалентно [27] максимизации следующего выражения:

$$\log p(x|z) - KL(q(z|x)||p(z))$$

Первый член представляет собой вероятность реконструкции, а второй член гарантирует, что наше изученное распределение q похоже на истинное априорное распределение p (расхождение равно 0, если распределения совпадают).

Чтобы вернуться к нашей графической модели, мы можем использовать q, чтобы вывести возможные скрытые переменные, которые использовались для генерации наблюдения x (рис.16).

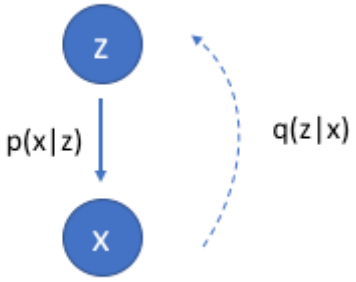

Рис.16. Цель – из наблюдения x получить скрытый параметр z [26].

Далее мы можем построить (представить) эту модель

в архитектуре нейронной сети, где модель кодировщика изучает отображение из *x* в *z*, а модель декодера изучает отображение из *z* обратно в *x*.

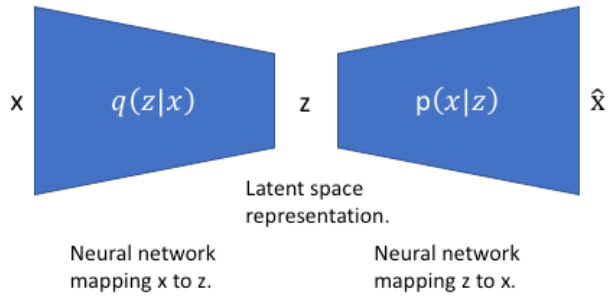

Рис. 17. Кодирование – декодирование [26]

Наша функция потерь для этой сети будет состоять из двух членов: один, который штрафует ошибку реконструкции (который можно рассматривать как максимизацию вероятности реконструкции, как обсуждалось ранее), и второй член, который поощряет наше изученное распределение *q(z|x)*, быть похожим на истинное априорное распределение *p(z)*, которое, как мы предположим, является распределением Гаусса, для каждого измерения j скрытого пространства:

$$L(x,\hat{x}) + \sum_i KL(q_i(z \mid x) \,||\, p(z))$$

Для чего необходимо изучать эти модели в курсе по кибербезопасности систем искусственного интеллекта? Для понимания их внутреннего устройства и, соответственно, для понимания того, как на модели можно воздействовать (как их атаковать). Вариационный кодировщик основан на формальном отображении реальных объектов (например, изображений) на некоторое пространство признаков, меньшего размера. А что если разные изображения будут иметь одинаковое отображение? Очевидно, декодер восстановит неверное изображение.

Пример работы, которая описывает атаку на вариационный кодировщик [28]:

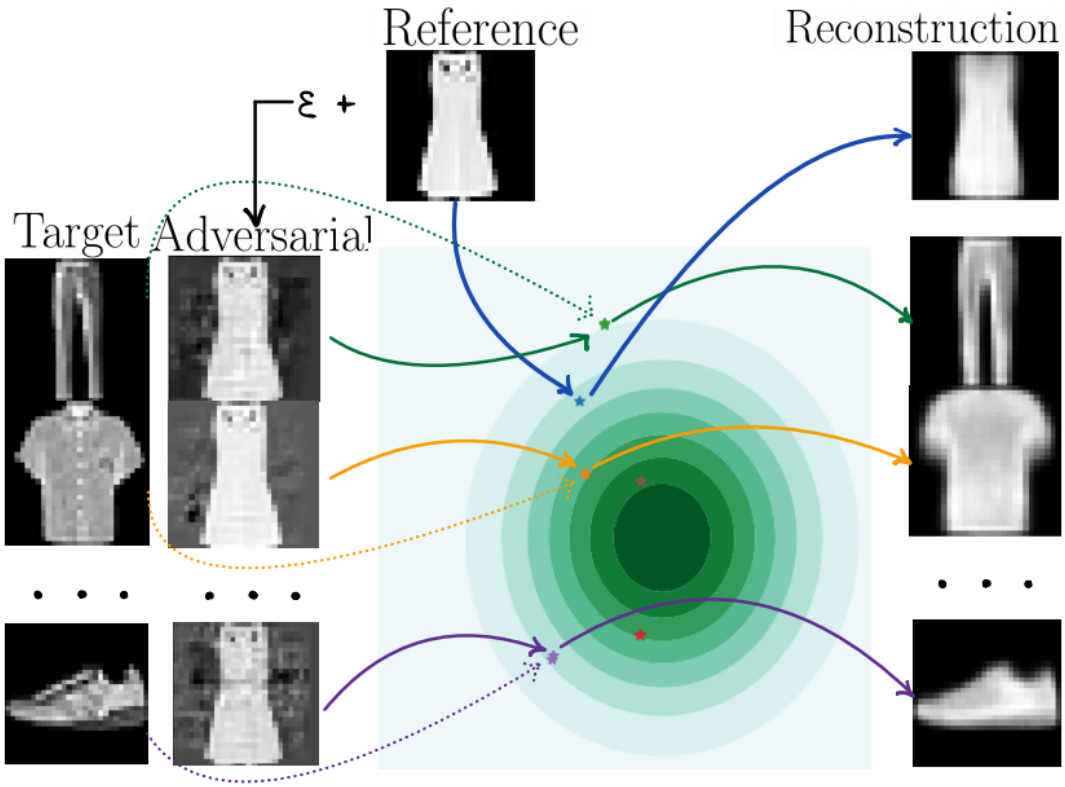

Рис. 18. Атака на автокодировщик [28].

Реальные изображения (первый столбец слева) и базовая картинка. Все они имеют свои собственные отображения (пунктирная линия) на пространство признаков и правильно восстанавливаются декодером. Базовое изображение (Reference) также имеет свое собственное отображение (синие линии). К базовому изображению добавляется некоторое возмущение ε, так что измененное базовое изображение (второй столбец слева – adversarial image) получает то же самое отображение в скрытом пространстве, что и реальное изображение в первом столбце (target).

IX GAN

Generative adversarial network [29] – возможно, наиболее известные (или широко используемые) в

настоящее время генеративные модели. Представляют особый интерес и с точки зрения безопасности систем машинного обучения, поскольку их алгоритм (схема работы), фактически, описывает прямую атаку – генерацию данных, похожих на реальные, но таковыми не являющимися.

Генеративно-состязательная нейросеть (*Generative adversarial network - GAN*) – это архитектура, состоящая из двух сетей: генеративной (генератора) и дискриминантной (дискриминатора), настроенных на работу друг против друга. Одна нейронная сеть (генератор) генерирует новые экземпляры данных, а другая (дискриминатор), оценивает их на подлинность; т.е. дискриминатор решает, относится ли каждый экземпляр данных, который он рассматривает, к набору тренировочных данных или нет.

Рис. 19. Реалистичные изображения, созданные с помощью GAN [30].

Дискриминантная сеть, анализируя образцы из оригинальных данных и из подделанных генератором, достигает некоторой точности различения. Генератор при этом начинает со случайных комбинаций параметров латентного пространства, а после оценки полученных образцов дискриминатором, применяется метод обратного распространения ошибки, который позволяет улучшить качество генерации, подправляя

Используя набор переменных латентного пространства, генеративная сеть пытается создать новый образец, смешав несколько исходных образцов. Дискриминантная сеть обучается различать подлинные и поддельные образцы, а результаты различения подаются на вход генеративной сети так, чтобы она смогла подобрать лучший набор латентных параметров, и дискриминантная сеть уже не смогла бы отличить подлинные образцы от поддельных. Таким образом, целью генератора является стремление повысить процент ошибок дискриминатора, а целью дискриминатора, наоборот, является улучшение точности распознавания.

входной набор скрытых параметров. В процессе совместного конкурентного обучения достигается минимаксное состояние равновесия, в котором обе сети значительно улучшили своё качество. Искусственные объекты (например, изображения) на выходе генеративной сети становятся всё более качественными (неотличимыми для дискриминантной сети от реально существующих).

Оригинальная архитектура GAN представлена на рисунке 20.

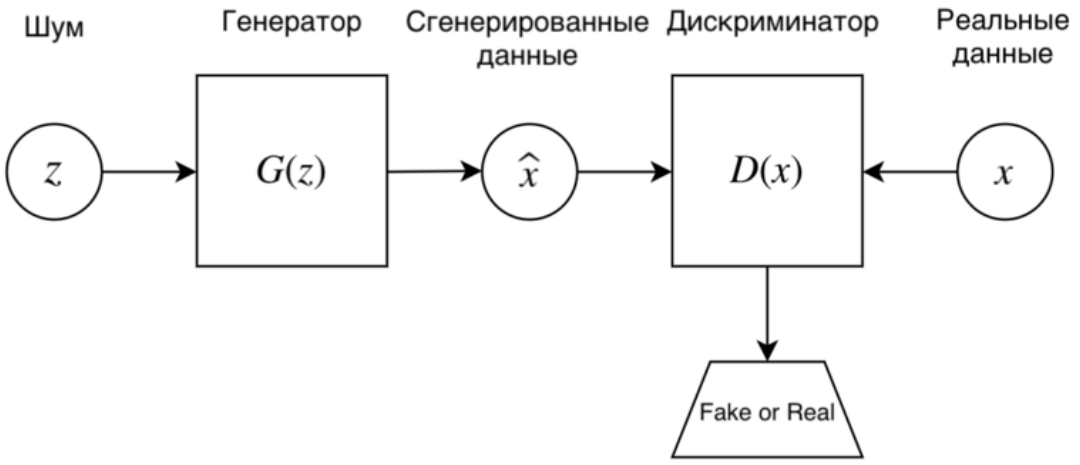

Рис. 20. Архитектура GAN [31]

Генераторная сеть непосредственно порождает примеры $x = g(z; \theta(g))$. Ее противник, дискриминантная сеть, пытается отличить примеры, взятые из обучающих данных, от примеров, порожденных генератором. Дискриминатор выдает значение, возвращенное функцией $d(x; \theta(d))$, равное вероятности того, что x – реальный обучающий пример, а не фальшивка, выбранная из модели. Описать процесс обучения в порождающей состязательной сети проще всего как игру с нулевой суммой, в которой функция $v(\theta(g), \theta(d))$ определяет платеж дискриминатора. Генератор получает $-v(\theta(g), \theta(d))$ в качестве своего платежа. В процессе обучения каждый игрок стремится максимизировать свой платеж, так что в пределе получаем

$$\arg \min_{g} \max_{d} v(d, g)$$

Практически, работа с изображениями, выглядит так [1]:

Генератор (нейронная сеть) последовательно создает изображения, которые оценивает дискриминатор

Рис.21. Последовательное создание изображений [1]

В процессе работы изображения изменяются с учетом оценки дискриминатора (рис. 22)

Рис. 22. Модификация созданных изображений [1]

До тех пор, пока не станут неотличимы (рис. 23)

Рис. 23. Остановка процесса [1]

Дискриминатор решает задачу классификации (рис.24). Соответственно, мы должны использовать две функции потерь при тренировке. Во время обучения дискриминатора он игнорирует потери в генераторе и просто использует свою функцию потерь. Потери генератора используются во время обучения генератора.

Дискриминатор классифицирует как реальные данные, так и поддельные данные от генератора.

Функция потерь дискриминатора наказывает дискриминатор за неправильную классификацию реального экземпляра как фальшивого или фальшивого экземпляра как реального.

Дискриминатор обновляет свои веса посредством обратного распространения потерь (желтый прямоугольник на рисунке 24)

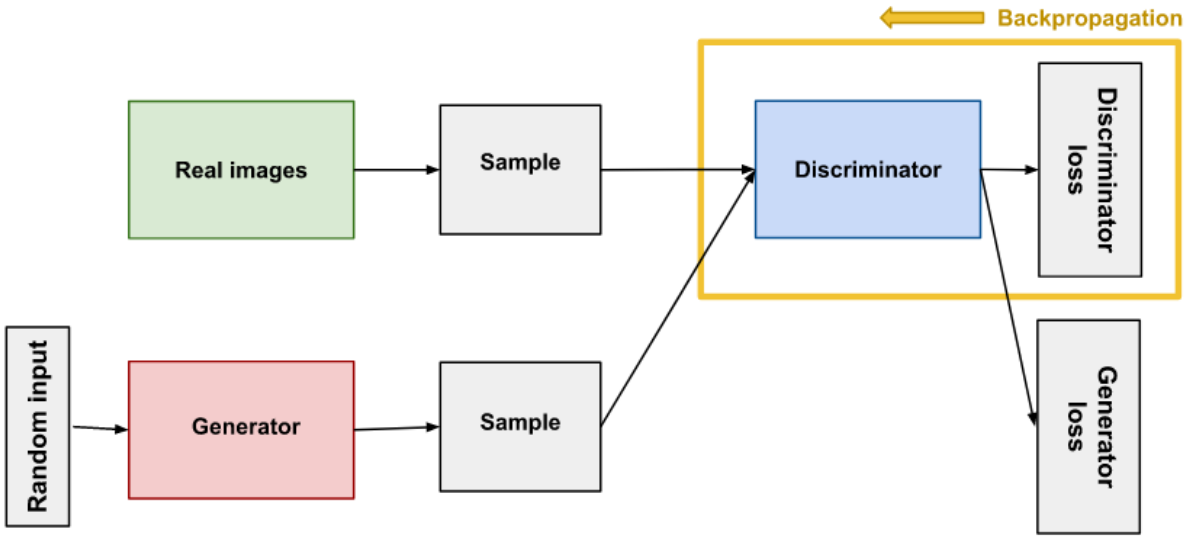

Рис. 24. Дискриминатор [1]

У генератора своя программа обучения, которая наказывает его за неудачную попытку обмануть дискриминатор

Рис. 25. Генератор [1]

В своей основной форме GAN принимает на вход случайный шум. Затем генератор преобразует этот шум в значимый выходной сигнал. Вводя шум, мы можем заставить GAN генерировать самые разные данные, выбирая их из разных мест в целевом распределении. Распределение этого шума не имеет большого значения, поэтому мы можем выбрать, для простоты, например, равномерное распределение. Для удобства пространство входных данных обычно имеет меньшую размерность, чем размерность выходного пространства.

При обучении генератора дискриминатор, обычно, не меняется. Обратное распространение на рисунке выше использует градиенты только для изменения весов в генераторе. Это общая идея – генератор не меняется во время тренировки дискриминатора, дискриминатор не меняется во время тренировки генератора. Соответственно, обучение GAN проходит в режиме чередования:

1. Дискриминатор тренируется для одной или нескольких эпох.
2. Генератор тренируется для одной или нескольких эпох.
3. Шаги 1 и 2 повторяются.

Тот факт, что тренируются две сети, создает, естественно, проблемы с совмещением их результатов. По мере того, как генератор улучшается с обучением, производительность дискриминатора ухудшается, потому что дискриминатор не может легко отличить реальные данные от фальшивых. Если генератор работает безупречно, то точность дискриминатора составляет 50% (по сути – подбрасывание монеты). То есть, обратная связь дискриминатора со временем становится менее значимой. Если GAN продолжает обучение после точки, когда дискриминатор выдает полностью случайную обратную связь, тогда генератор начинает тренироваться на нежелательной обратной связи, и его собственное качество может ухудшиться.

Цитата из Google: “Для GAN конвергенция часто является мимолетным, а не стабильным состоянием”. Что, между прочим, является хорошей базой для атак на GAN-модели.

GAN пытаются воспроизвести распределение вероятностей. Поэтому функции потерь, отражают расстояние между распределением данных, генерируемых GAN, и распределением реальных данных. Функции потерь две - одна для обучения генератора и одна для обучения дискриминатора. В

оригинальной схеме GAN потери генератора и дискриминатора зависят от единственной меры расстояния между распределениями вероятностей. При этом генератор может воздействовать только на распределение фальшивых данных. А при обучении дискриминатора мы можем также учитывать распределение реальных данных.

Следующая картинка из оригинальной статьи Goodfellow (изобретателя GAN) как раз и описывает стохастические градиенты в функциях потерь для дискриминатора и генератора (подчеркнуты, соответственно, красным и зеленым).

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Рис. 26. Стохастические градиенты в GAN [32]

log(D(x)) относится к вероятности того, что дискриминатор правильно классифицирует реальное изображение, максимизация *log(1-D(G(z)))* поможет ему правильно пометить поддельное изображение, которое исходит от генератора.

Есть различные варианты GAN, например, Wasserstein Generative Adversarial Network (WGAN) [33] или Conditional Generative Adversarial Network (CGAN) [34]. Они отличаются как раз функциями потерь. В различных вариантах GAN также входные данные генератора могут выбираться специальным образом.

Благодарности

Настоящая работа подготовлена на кафедре Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова в рамках развития магистерской программы "Искусственный интеллект в кибербезопасности" [35].

Статья является продолжением серии публикаций, посвященных устойчивым моделям машинного обучения [36, 37, 38].

Работа подготовлена при поддержке Междисциплинарной научно-образовательной школы Московского университета «Мозг, когнитивные системы, искусственный интеллект».

Библиография

[1] Generative Adversarial Networks https://developers.google.com/machine-learning/gan Retrieved: May, 2022

[2] Nalisnick, Eric, et al. "Do deep generative models know what they don't know?." arXiv preprint arXiv:1810.09136 (2018).

[3] Oussidi, Achraf, and Azeddine Elhassouny. "Deep generative models: Survey." 2018 International Conference on Intelligent Systems and Computer Vision (ISCV). IEEE, 2018.

[4] Goodfellow, Ian. "Nips 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).

[5] Reynolds, Douglas A. "Gaussian mixture models." Encyclopedia of biometrics 741.659-663 (2009).

[6] Mixture models https://en.wikipedia.org/wiki/Mixture_model Retrieved: May, 2022

[7] Gaussian Mixture Model https://www.kaggle.com/code/dfoly1/gaussian-mixture-model/notebook Retrieved: May, 2022

[8] Build Better and Accurate Clusters with Gaussian Mixture Models https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/ Retrieved: May, 2022

[9] In Depth: Gaussian Mixture Models https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html Retrieved: May, 2022

[10] Eddy, Sean R. "What is a hidden Markov model?." Nature biotechnology 22.10 (2004): 1315-1316.

[11] Next Word Prediction using Markov Model https://www.kaggle.com/discussions/getting-started/107497 Retrieved: May, 2022

[12] Скрытая марковская модель https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D1%8B%D1%82%D0%B0%D1%8F_%D0%BC%D0%B0%D1%80%D0%B

A%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C Retrieved: May, 2022
[13] Markov and Hidden Markov Model https://towardsdatascience.com/markov-and-hidden-markov-model-3eec42298d75 Retrieved: May, 2022
[14] Jelinek, Frederick, John D. Lafferty, and Robert L. Mercer. "Basic methods of probabilistic context free grammars." Speech Recognition and Understanding. Springer, Berlin, Heidelberg, 1992. 345-360.
[15] Raghavan, Sindhu, Adriana Kovashka, and Raymond Mooney. "Authorship attribution using probabilistic context-free grammars." Proceedings of the ACL 2010 conference short papers. 2010.
[16] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.
[17] Latent Dirichlet Allocation https://devopedia.org/latent-dirichlet-allocation Retrieved: May, 2022
[18] Chen, Serena H., and Carmel A. Pollino. "Good practice in Bayesian network modelling." Environmental Modelling & Software 37 (2012): 134-145.
[19] Nikkarila, Juha-Pekka, Ilmari Kangasniemi, and Janne Valtonen. "Bayesian networks: an example of software and some defence applications." (2015).
[20] Heckerman, David. "A Bayesian approach to learning causal networks." arXiv preprint arXiv:1302.4958 (2013).
[21] Hinton, Geoffrey E. "Boltzmann machine." Scholarpedia 2.5 (2007): 1668.
[22] Boltzmann machine https://en.wikipedia.org/wiki/Boltzmann_machine Retrieved: May, 2022
[23] Book-Recommender-System-RBM https://github.com/adityashrm21/book-recommender-system-rbm Retrieved: May, 2022
[24] Understanding Variational Autoencoders (VAEs) from two perspectives: deep learning and graphical models https://jaan.io/what-is-variational-autoencoder-vae-tutorial/ Retrieved: May, 2022
[25] Generative modelling using Variational AutoEncoders(VAE) and Beta-VAE's https://medium.com/analytics-vidhya/generative-modelling-using-variational-autoencoders-vae-and-beta-vaes-81a56ef0bc9f Retrieved: May, 2022
[26] Variational autoencoders https://www.jeremyjordan.me/variational-autoencoders/ Retrieved: May, 2022
[27] Ghojogh, Benyamin, et al. "Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey." arXiv preprint arXiv:2101.00734 (2021).
[28] Kuzina, Anna, Max Welling, and Jakub M. Tomczak. "Diagnosing vulnerability of variational auto-encoders to adversarial attacks." arXiv preprint arXiv:2103.06701 (2021).
[29] Creswell, Antonia, et al. "Generative adversarial networks: An overview." IEEE Signal Processing Magazine 35.1 (2018): 53-65.
[30] Faster Learning and Better Image Quality with Evolving Generative Adversarial Networks https://neurohive.io/en/state-of-the-art/faster-learning-and-better-image-quality-with-evolving-generative-adversarial-networks/ Retrieved: May, 2022
[31] Generative Adversarial Nets (GAN) https://neerc.ifmo.ru/wiki/index.php?title=Generative_Adversarial_Nets_(GAN) Retrieved: May, 2022
[32] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[33] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.
[34] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
[35] Artificial Intelligence in Cybersecurity. http://master.cmc.msu.ru/?q=ru/node/3496 (in Russian) Retrieved: Dec, 2021
[36] Намиот Д.Е., Ильюшин Е.А., Чижов И.В. Текущие академические и индустриальные проекты, посвященные устойчивому машинному обучению //International Journal of Open Information Technologies. – 2021. – Т. 9. – No. 10. – С. 35-46.
[37] Намиот, Д. Е., Е. А. Ильюшин, and И. В. Чижов. "АТАКИ НА СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ-ОБЩИЕ ПРОБЛЕМЫ И МЕТОДЫ." International Journal of Open Information Technologies 10.3 (2022): 17-22.
[38] Намиот Д. Е., Ильюшин Е. А., Чижов И. В. Основания для работ по устойчивому машинному обучению //International Journal of Open Information Technologies. – 2021. – Т. 9. – №. 11. – С. 68-74.

Generative Models in Machine Learning

Dmitry Namiot, Eugene Ilyushin

Abstract— **This article, written for the Robust Machine Learning Curriculum, discusses the so-called Generative Models in Machine Learning. Generative models learn the distribution of data from some sample data set and then can generate (create) new data instances. Generative models are popular tools with a wide range of applications. Recent advances in deep learning have led to improvements in the architecture of generative models, and some current models can (in some cases) produce realistic enough results to fool both end-users (humans) and recognition and classification algorithms. Generative models are used in constructing adversarial attacks. Instead of looking for minimal modifications, as in classical evasion attacks, generator models allow, for example, to create adversarial examples completely from scratch. At the same time, generator models are just as vulnerable to adversarial attacks as classifiers. This is our first material on this topic and, obviously, consideration of this important topic will be continued.**

Keywords— **generative models, machine learning.**

REFERENCES

[1] Generative Adversarial Networks https://developers.google.com/machine-learning/gan Retrieved: May, 2022
[2] Nalisnick, Eric, et al. "Do deep generative models know what they don't know?." arXiv preprint arXiv:1810.09136 (2018).
[3] Oussidi, Achraf, and Azeddine Elhassouny. "Deep generative models: Survey." 2018 International Conference on Intelligent Systems and Computer Vision (ISCV). IEEE, 2018.
[4] Goodfellow, Ian. "Nips 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).
[5] Reynolds, Douglas A. "Gaussian mixture models." Encyclopedia of biometrics 741.659-663 (2009).
[6] Mixture models https://en.wikipedia.org/wiki/Mixture_model Retrieved: May, 2022
[7] Gaussian Mixture Model https://www.kaggle.com/code/dfoly1/gaussian-mixture-model/notebook Retrieved: May, 2022
[8] Build Better and Accurate Clusters with Gaussian Mixture Models https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/ Retrieved: May, 2022
[9] In Depth: Gaussian Mixture Models https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html Retrieved: May, 2022
[10] Eddy, Sean R. "What is a hidden Markov model?." Nature biotechnology 22.10 (2004): 1315-1316.
[11] Next Word Prediction using Markov Model https://www.kaggle.com/discussions/getting-started/107497 Retrieved: May, 2022
[12] Skrytaja markovskaja model' https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D1%8B%D1%82%D0%B0%D1%8F_%D0%BC%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C Retrieved: May, 2022
[13] Markov and Hidden Markov Model https://towardsdatascience.com/markov-and-hidden-markov-model-3eec42298d75 Retrieved: May, 2022
[14] Jelinek, Frederick, John D. Lafferty, and Robert L. Mercer. "Basic methods of probabilistic context free grammars." Speech Recognition and Understanding. Springer, Berlin, Heidelberg, 1992. 345-360.
[15] Raghavan, Sindhu, Adriana Kovashka, and Raymond Mooney. "Authorship attribution using probabilistic context-free grammars." Proceedings of the ACL 2010 conference short papers. 2010.
[16] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.
[17] Latent Dirichlet Allocation https://devopedia.org/latent-dirichlet-allocation Retrieved: May, 2022
[18] Chen, Serena H., and Carmel A. Pollino. "Good practice in Bayesian network modelling." Environmental Modelling & Software 37 (2012): 134-145.
[19] Nikkarila, Juha-Pekka, Ilmari Kangasniemi, and Janne Valtonen. "Bayesian networks: an example of software and some defence applications." (2015).
[20] Heckerman, David. "A Bayesian approach to learning causal networks." arXiv preprint arXiv:1302.4958 (2013).
[21] Hinton, Geoffrey E. "Boltzmann machine." Scholarpedia 2.5 (2007): 1668.
[22] Boltzmann machine https://en.wikipedia.org/wiki/Boltzmann_machine Retrieved: May, 2022
[23] Book-Recommender-System-RBM https://github.com/adityashrm21/book-recommender-system-rbm Retrieved: May, 2022
[24] Understanding Variational Autoencoders (VAEs) from two perspectives: deep learning and graphical models https://jaan.io/what-is-variational-autoencoder-vae-tutorial/ Retrieved: May, 2022
[25] Generative modelling using Variational AutoEncoders(VAE) and Beta-VAE's https://medium.com/analytics-vidhya/generative-modelling-using-variational-autoencoders-vae-and-beta-vaes-81a56ef0bc9f Retrieved: May, 2022
[26] Variational autoencoders https://www.jeremyjordan.me/variational-autoencoders/ Retrieved: May, 2022
[27] Ghojogh, Benyamin, et al. "Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey." arXiv preprint arXiv:2101.00734 (2021).
[28] Kuzina, Anna, Max Welling, and Jakub M. Tomczak. "Diagnosing vulnerability of variational auto-encoders to adversarial attacks." arXiv preprint arXiv:2103.06701 (2021).
[29] Creswell, Antonia, et al. "Generative adversarial networks: An overview." IEEE Signal Processing Magazine 35.1 (2018): 53-65.
[30] Faster Learning and Better Image Quality with Evolving Generative Adversarial Networks https://neurohive.io/en/state-of-the-art/faster-learning-and-better-image-quality-with-evolving-generative-adversarial-networks/ Retrieved: May, 2022
[31] Generative Adversarial Nets (GAN) https://neerc.ifmo.ru/wiki/index.php?title=Generative_Adversarial_Nets_(GAN) Retrieved: May, 2022
[32] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[33] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.
[34] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
[35] Artificial Intelligence in Cybersecurity. http://master.cmc.msu.ru/?q=ru/node/3496 (in Russian) Retrieved: Dec, 2021
[36] Namiot D.E., Il'jushin E.A., Chizhov I.V. Tekushhie akademicheskie i industrial'nye proekty, posvjashhennye ustojchivomu mashinnomu obucheniju //International Journal of Open Information Technologies. – 2021. – T. 9. – No. 10. – S. 35-46.
[37] Namiot, D. E., E. A. Il'jushin, and I. V. Chizhov. "ATAKI NA SISTEMY MAShINNOGO OBUChENIJa-OBShhIE PROBLEMY I METODY." International Journal of Open Information Technologies 10.3 (2022): 17-22.
[38] Namiot D. E., Il'jushin E. A., Chizhov I. V. Osnovanija dlja rabot po ustojchivomu mashinnomu obucheniju //International Journal of Open Information Technologies. – 2021. – T. 9. – #. 11. – S. 68-74.