ЛЕКЦИЯ 5. Ориентированные порождающие сети

Ориентированные графические модели составляют важный класс графических моделей. Но, несмотря на их популярность в широком сообществе машинного обучения, в более узком кругу специалистов по глубокому обучению примерно до 2013 года их затмевали неориентированные модели типа ОМБ.

Рассмотрим некоторые стандартные графические модели, которые традиционно ассоциируются с глубоким обучением. Мы уже описывали глубокие сети доверия, представляющие собой частично ориентированную модель. Также описывали модели разреженного кодирования, которые можно считать мелкими ориентированными порождающими моделями. В контексте глубокого обучения они часто используются для обучения признакам, хотя оставляют желать лучшего как метод генерации примеров и оценивания плотности. Теперь опишем различные виды глубоких полностью ориентированных моделей.

Сигмоидные сети доверия

Сигмоидная сеть доверия – простой вид ориентированной графической модели со специфическим условным распределением вероятности. В общем случае такую сеть можно представлять себе как вектор бинарных состояний s, каждый элемент которого зависит от своих предков:

$$p(s_i) = \sigma\left(\sum_{j<i} W_{j,i}\, s_j + b_i\right) \quad (1)$$

В самом распространенном случае сигмоидная сеть доверия состоит из большого числа слоев, а предковая выборка проходит через много скрытых слоев и в конечном итоге генерирует видимый слой. Эта структура очень похожа на глубокую сеть доверия – с тем отличием, что блоки в начале процесса выборки независимы друг от друга, а не выбираются из ограниченной машины Больцмана. Такая структура интересна по целому ряду причин, в частности потому, что является универсальным аппроксиматором распределений вероятности видимых блоков в том смысле, что может аппроксимировать любое распределение вероятности бинарных величин с произвольной точностью при наличии достаточно большого числа слоев, даже если ширина каждого слоя ограничена размерностью видимого слоя.

Хотя в сигмоидной сети доверия генерация выборки видимых слоев очень эффективна, о большинстве других операций этого не скажешь. Вывод скрытых блоков приусловии видимых блоков вычислительно неразрешим. Как и вывод среднего поля, поскольку для вычисления вариационной нижней границы нужно знать математические ожидания клик, а они охватывают слои целиком. Из-за трудности этой проблемы ориентированные дискретные сети не получили широкого распространения.

Один из подходов к выводу в сигмоидной сети доверия – построить другую нижнюю границу специально для таких сетей. Но этот подход применялся только к совсем небольшим сетям. Другое решение – воспользоваться механизмами обученного вывода.

Дифференцируемые генераторные сети

В основе многих порождающих моделей лежит идея использования дифференцируемой генераторной сети. Модель преобразует примеры латентных переменных z в примеры x или в распределения примеров x, применяя дифференцируемую функцию $g(z;\theta^{(g)})$, которая обычно представляется нейронной сетью. В этот класс моделей входят автокодировщики, которые объединяют генераторную сеть с сетью вывода, порождающие состязательные сети, которые объединяют генераторную сеть с дискриминантной, и методы, в которых генераторные сети используются сами по себе.

По сути своей генераторные сети – это просто параметризованные вычислительные процедуры для генерации примеров, где архитектура предоставляет семейство распределений, из которых можно производить выборку, а с помощью параметров выбирается конкретное распределение из этого семейства.

Например, стандартная процедура выборки из нормального распределения со средним μ и ковариационной матрицей Σ заключается в том, чтобы подать выборку z из нормального распределения с нулевым средним и единичной ковариационной матрицей на вход очень простой генераторной сети, которая содержит всего один аффинный слой:

$$x = g(z) = \mu + Lz, \tag{2}$$

где L определяется разложением Холеского матрицы Σ.

Разложение Холецкого (метод квадратного корня) – представление симметричной положительно определённой матрицы $A = LL^{\top}$, где L – нижняя треугольная матрица со строго положительными элементами на диагонали, а $L^{\top}$ – верхняя треугольная матрица.

Генераторы псевдослучайных чисел также могут использовать нелинейные преобразования простых распределений. Например, в методе обратного преобразования выбирается скаляр z из распределения $U(0,1)$ и применяется нелинейное преобразование к скаляру x. В этом случае $g(z)$ определяется как обращение интегральной функции распределения $F(x) = \int_{-\infty}^{x} p(v)dv$. Если мы умеем задавать $p(x)$, интегрировать по x и обращать получающуюся функцию, то сможем произвести выборку из $p(x)$ без применения машинного обучения.

Чтобы сгенерировать примеры из более сложных распределений, которые трудно описать непосредственно, трудно проинтегрировать или трудно обратить результат интегрирования, мы пользуемся сетью прямого распространения для представления параметрического семейства нелинейных функций g и с помощью обучающих данных выводим параметры, отбирающие нужную функцию.

Можно считать, что g задает нелинейную замену переменных, преобразующую распределение z в желаемое распределение x.

Для обратимой дифференцируемой непрерывной функции g имеет место тождество

$$p_z(z) = p_x\big(g(z)\big)\left|det\left(\frac{\partial g}{\partial z}\right)\right| \tag{3}$$

где $det\left(\frac{\partial g}{\partial z}\right)$ – определитель $\left(\frac{\partial g}{\partial z}\right)$

Тем самым мы неявно определяем распределение вероятности x:

$$p_x(x) = \frac{p_z\left(g^{-1}(x)\right)}{\left|det\left(\frac{\partial g}{\partial z}\right)\right|} \tag{4}$$

Разумеется, при некоторых g это выражение трудно вычислить, поэтому часто применяются непрямые методы обучения g, вместо того чтобы пытаться максимизировать $\log p(x)$ непосредственно.

В некоторых случаях используется g не для получения примера x напрямую, а для определения условного распределения x. Например, можно было бы воспользоваться генераторной сетью, последний слой которой состоит из сигмоидных выходов, для получения параметров распределений Бернулли:

$$p(x_i = 1|z) = g(z)_i \tag{5}$$

В этом случае, используя g для определения $p(x|z)$, определяем распределение x путем исключения z:

$$p(x) = \mathbb{E}_z p(x|z) \tag{6}$$

Оба подхода определяют распределение $p_g(x)$ и позволяют обучать различные критерии p_g, используя технику перепараметризации.

У двух разных подходов к определению генераторных сетей – порождение параметров условного распределения и прямое порождение примеров – есть свои плюсы и минусы. Если генераторная сеть определяет условное распределение x, то она способна генерировать как дискретные, так и непрерывные данные. Если же генераторная сеть порождает примеры непосредственно, то она может генерировать только непрерывные данные (можно было бы включить дискретизацию в прямое распространение, но тогда модель нельзя было бы обучить с помощью обратного распространения). Преимущество

непосредственной выборки – в том, что мы больше не ограничены просто записываемыми условными распределениями, к которым проектировщик мог бы легко применять алгебраические преобразования.

В обоснование подходов, основанных на дифференцируемых генераторных сетях, выдвигается успешное применение градиентного спуска к дифференцируемым сетям прямого распространения для целей классификации. В контексте обучения с учителем глубокие сети прямого распространения, обученные градиентными методами, практически наверняка приводят к успеху при наличии достаточного числа скрытых блоков и обучающих данных. Нельзя ли перенести тот же рецепт успеха на порождающее моделирование?

Порождающее моделирование выглядит труднее классификации или регрессии, потому что процесс обучения требует оптимизации неразрешимых критериев. В дифференцируемых генераторных сетях критерии неразрешимы, потому что данные не содержат одновременно входов z и выходов x генераторной сети. В случае обучения с учителем задавались входы x и выходы y, а процедуре оптимизации нужно было только обучиться, как порождать заданное отображение. В случае порождающего моделирования процедура обучения должна определить, как организовать пространство z полезным способом и, кроме того, как отобразить z в x.

Рассмотрим простую задачу. Пусть соответствие между z и x задано. Точнее говоря, обучающие данные представляют собой сгенерированные компьютером изображения стула. Латентные переменные z – параметры движка отрисовки, определяющие выбор модели стула, его положение и другие детали, влияющие на генерацию изображения. Используя эти синтетические данные, сверточная сеть может обучиться отображать z (описания содержимого изображения) в x (аппроксимации отрисованных изображений). Это наводит на мысль, что современные дифференцируемые генераторные сети обладают достаточной емкостью, чтобы стать хорошими порождающими моделями, и что современные алгоритмы оптимизации вполне способны их аппроксимировать. Трудность заключается в том, как обучать генераторные сети, если значение z, соответствующее каждому x, не фиксировано и заранее неизвестно.

Теперь рассмотрим несколько подходов к обучению дифференцируемых генераторных сетей при наличии только обучающих примеров x.

Вариационные автокодировщики

Вариационный автокодировщик, или VAE, – это ориентированная модель, в которой применяется обученный приближенный вывод и которую можно обучить с помощью одних лишь градиентных методов.

Для порождения выборки из модели VAE сначала выбирает пример z из кодового распределения $p_{model}(z)$. Затем этот пример прогоняется через дифференцируемую генераторную сеть $g(z)$. Наконец, производится выборка x из распределения $p_{model}(x, g(z)) = p_{model}(x|z)$. Но на этапе обучения для получения z используется сеть приближенного вывода (или кодировщик) $q(z|x)$, и тогда $p_{model}(x|z)$ рассматривается как декодирующая сеть.

Главная идея вариационных автокодировщиков заключается в том, что их можно обучить с помощью максимизации вариационной нижней границы $\mathcal{L}(q)$, ассоциированной с точкой x:

$$\mathcal{L}(q) = \mathbb{E}_{z\sim q(z|x)}\, log\, p_{model}(z, x) + \mathcal{H}(q(z|x)) =$$

$$\mathbb{E}_{z\sim q(z|x)}\, log\, p_{model}(x|z) - D_{KL}(q(z|x)||\, p_{model}(z)) \leq \log p_{model}(x) \qquad (7)$$

В равенстве (7) первый член – не что иное, как совместное логарифмическое правдоподобие видимых и латентных переменных относительно приближенного апостериорного распределения латентных переменных. Если в качестве q выбрано нормальное распределение с шумом, добавленным к предсказанному среднему значению, то максимизация этого энтропийного члена поощряет увеличение стандартного отклонения шума. В общем случае энтропийный член поощряет вариационное апостериорное распределение отдавать больше массы вероятности многим значениям z, которые могли бы породить x, а не сосредоточивать ее в одной точечной оценке наиболее вероятного значения. Далее в равенстве (7) в первом члене легко распознать логарифмическое правдоподобие реконструкции, встречающееся и в других автокодировщиках. Второй член пытается сблизить приближенное апостериорное распределение $q(z|x)$ и априорное модельное распределение $p_{model}(z)$.

В традиционных подходах к вариационному выводу и обучению q выводится с помощью алгоритма оптимизации. Это медленно и зачастую требует умения вычислять математическое ожидание $\mathbb{E}_{z\sim q}\, log\, p_{model}(z, x)$ в замкнутой форме. Основная идея вариационного автокодировщика – обучить параметрический кодировщик (который иногда называют сетью вывода или моделью распознавания), порождающий параметры q. Если z – непрерывная переменная, то мы тогда сможем выполнить обратное распространение через примеры z, выбранные из $q(z|x) = q(z; f(x; \theta))$, для получения градиента по θ. В таком случае обучение состоит просто из максимизации $\mathcal{L}$ относительно параметров кодировщика и декодера. Все математические ожидания в $\mathcal{L}$ можно аппроксимировать с помощью выборки методом Монте-Карло.

Подход на основе вариационного автокодировщика элегантный, теоретически удовлетворительный и простой в реализации. Ко всему прочему, он дает отличные результаты и входит в число передовых подходов к порождающему моделированию. Главный недостаток заключается в том, что выборки из вариационных автокодировщиков, обученных на изображениях, получаются несколько размытыми. Причина этого феномена до сих пор неясна. Возможно, что размытость – свойство, внутренне присущее критерию максимального правдоподобия, по которому минимизируется $D_{KL}(q(z|x)||\, p_{model}(z))$ – расхождение Кульбака–Лейблера между $q(z|x)$ и $p_{model}(z)$. Первый аргумент функционала в расхождении Кульбака–Лейблера обычно интерпретируется как истинное или постулируемое априори распределение, второй – как предполагаемое (проверяемое). Как показано, на рис. 1. это означает, что модель назначает высокую вероятность точкам, которые встречаются в обучающем наборе, и, возможно, каким-то другим точкам. Вот эти другие точки и могут включать размытые изображения.

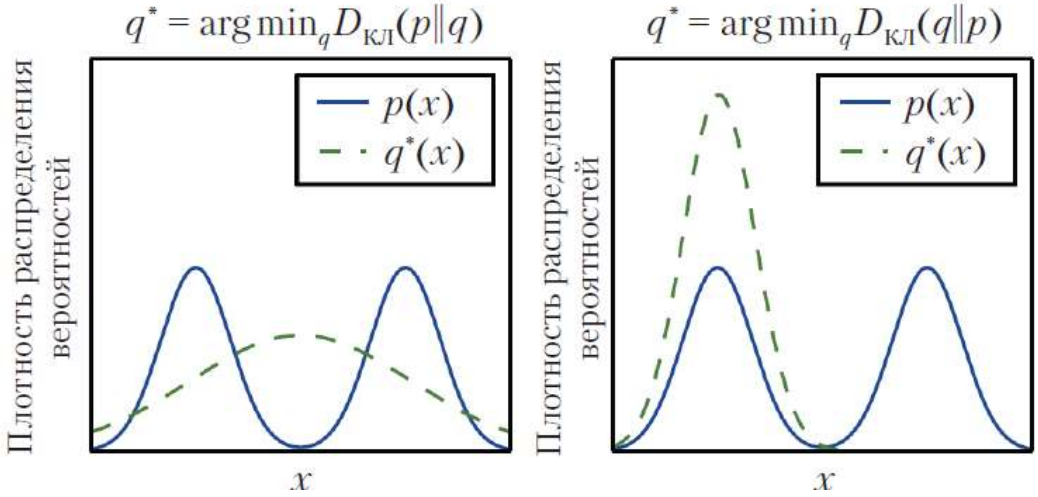

Рис. 1. Расхождение КЛ несимметрично. Предположим, что мы хотим аппроксимировать распределение $p(x)$ другим распределением $q(x)$. Можно выбирать, что минимизировать: $D_{KL}(p||q)$ или $D_{KL}(q||p)$. На рисунке показаны последствия выбора в случае, когда p – смесь двух нормальных распределений, а q – обычное нормальное распределение. Выбор направления расхождения зависит от задачи. Для одних приложений нужна аппроксимация, в которой вероятность высока там, где высока вероятность истинного распределения, а для других – чтобы была низкая вероятность там, где низка вероятность истинного распределения. (Слева) Результат минимизации $D_{KL}(p||q)$. В этом случае q выбирается так, чтобы была высокая вероятность там, где высока вероятность p. Если p имеет несколько мод, то q стремится размазать моды, собрав заключенную в них массу вероятности. (Справа) Результат минимизации $D_{KL}(q||p)$. В этом

случае q выбирается так, чтобы была низкая вероятность там, где низка вероятность p. Если p имеет несколько достаточно далеко отстоящих мод, как на этом рисунке, то расхождение КЛ (Кульбака–Лейблера) достигает минимума, когда выбирается одна мода, чтобы предотвратить размещение массы вероятности в областях низкой вероятности между модами p. На рисунке показан результат, когда q выбрано, так чтобы усилить левую моду. Такое же значение расхождения КЛ можно было бы получить, выбрав правую моду. Если моды не разделены достаточно выраженной областью малой вероятности, то и при таком выборе направления расхождения КЛ может произойти размазывание мод.

Таким образом, отчасти причина того, что модель предпочитает назначать большую массу вероятности размытым изображениям, а не каким-то другим частям пространства, состоит в том, что вариационные автокодировщики, применяемые на практике, обычно имеют нормальное распределение $p_{model}(x, g(z))$. Максимизация нижней границы правдоподобия такого распределения похожа на обучение традиционного автокодировщика по критерию среднеквадратической ошибки в том смысле, что склонна игнорировать признаки входа, которые занимают мало пикселей или приводят лишь к небольшому изменению яркости тех пикселей, которые занимают. Эта проблема характерна не только для VAE, но и для других порождающих моделей, которые оптимизируют логарифмическое правдоподобие или, что то же самое, $D_{KL}(p_{data}||p_{model})$. У современных моделей VAE есть еще одна неприятная проблема – они склонны использовать лишь малое подмножество измерений z, как будто кодировщик не способен преобразовать достаточно много локальных направлений в пространстве входов в пространство, где маргинальное распределение совпадает с факторизованным априорным распределением.

Инфраструктура VAE легко обобщается на разнообразные архитектуры моделей. Это важное преимущество, по сравнению с машинами Больцмана, которые требуют скрупулезно проектировать модель, чтобы избежать вычислительной неразрешимости. VAE прекрасно работают с широким семейством дифференцируемых операторов. Из особо изощренных VAE упомянем модель глубокого рекуррентного внимательного писателя (deep recurrent attentive writer – DRAW). В модели DRAW используются рекуррентный кодировщик и рекуррентный декодер в сочетании с механизмом внимания. Порождающий процесс в DRAW состоит из последовательного посещения небольших участков изображения и выборки значений пикселей в этих точках. VAE также можно обобщить на порождение последовательностей, если

определить вариационные РНС, используя рекуррентные кодировщик и декодер в составе инфраструктуры VAE. Выборка из традиционных РНС включает только недетерминированные операции в пространстве выходов. Вариационные РНС обладают также возможностью рандомизации на потенциально более абстрактном уровне, улавливаемом латентными переменными VAE.

Инфраструктура VAE обобщена также на максимизацию не только традиционной вариационной нижней границы, но и на целевую функцию автокодировщика, взвешенного по значимости:

$$\mathcal{L}_k(x, g) = \mathbb{E}_{z^{(1)},\dots,z^{(k)}\sim q(Z|X)}\left[log\,\frac{1}{k}\sum_{i=1}^{k}\frac{p_{model}(x,z^{(i)})}{q(z^{(i)}|x)}\right] \quad (8)$$

При $k = 1$ эта новая целевая функция эквивалентна традиционной нижней границе $\mathcal{L}$. Но ее можно также интерпретировать как оценку истинного $log\,p_{model}(x)$ с использованием выборки по значимости z из вспомогательного распределения $q(z|x)$. Кроме того, она является нижней границей $log\,p_{model}(x)$ и с увеличением k становится точнее.

У вариационных автокодировщиков есть много интересных связей с многопредсказательными глубокими машинами Больцмана (МП-ГМБ) и другими подходами, включающими обратное распространение по графу приближенного вывода. В предшествующих подходах требовалось, чтобы граф вычислений поставляла процедура вывода, например, решение уравнений неподвижной точки среднего поля. Вариационный автокодировщик определен для произвольных графов вычислений, поэтому применим к более широкому классу вероятностных моделей, т. к. необязательно ограничиваться лишь моделями, для которых разрешимы уравнения неподвижной точки среднего поля. У вариационного автокодировщика есть еще одно достоинство – он увеличивает границу логарифмического правдоподобия модели, тогда как критерии для МП-ГМБ и родственных моделей в большей степени эвристические и почти не допускают вероятностной интерпретации, а лишь обеспечивают верность результатов приближенного вывода. Недостаток же VAE в том, что он обучает сеть вывода решению только одной задачи: выводу z по заданному x. Более ранние методы способны выполнять приближенный вывод любого подмножества переменных по любому другому известному подмножеству, поскольку уравнения неподвижной точки среднего поля описывают, как разделяются параметры между графами вычислений для этих разных задач.

Полезное свойство вариационного автокодировщика состоит в том, что одновременное обучение параметрического кодировщика в сочетании с генераторной сетью побуждает модель обучиться предсказуемой системе

координат, которую может запомнить кодировщик. Благодаря этому VAE становится отличным алгоритмом обучения многообразий. На рис. 2 показаны примеры многообразий низкой размерности, обученных вариационным автокодировщиком. В одном из продемонстрированных на рисунке случаев алгоритм выявил два независимых фактора вариативности в изображениях лиц: угол поворота и эмоциональное выражение.

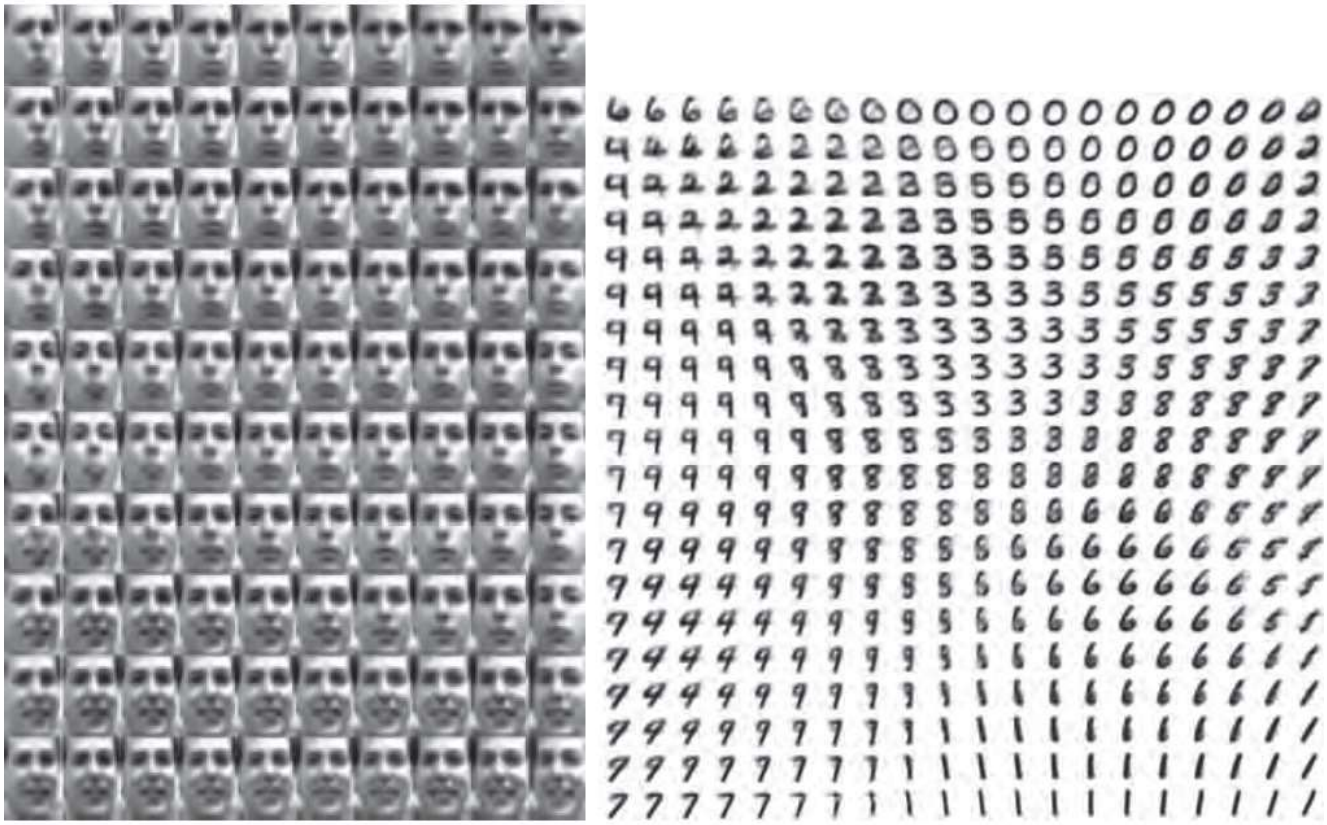

Рис. 2 Примеры двумерных систем координат для многообразий высокой размерности, обученных вариационным автокодировщиком. На странице можно нарисовать два измерения, поэтому мы можем составить некоторое представление о работе модели, обучив ее двумерному латентному коду, даже если полагаем, что истинная размерность многообразия данных гораздо выше. Показанные изображения – не примеры, взятые из обучающего набора, а изображения x, фактически сгенерированные моделью $p(x|z)$ путем простого изменения двумерного «кода» z (каждому изображению соответствует свой выбор «кода» z на двумерной равномерной сетке). (Слева) Двумерное отображение многообразия лиц Фрея. Одно выявленное измерение (по горизонтали) соответствует главным образом углу поворота лица, а другое (по вертикали) – эмоциональному выражению. (Справа) Двумерное отображение многообразия MNIST.

Порождающие состязательные сети

Порождающие состязательные сети, или ПСС, – еще один подход к порождающему моделированию, основанный на дифференцируемых генераторных сетях. В их основе лежит теоретико-игровая ситуация, когда генераторная сеть должна состязаться с противником. 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

$$g^* = \arg\min_g \max_d (g, d) \tag{9}$$

По умолчанию v выбирается следующим образом:

$$v\left(\theta^{(g)}, \theta^{(d)}\right) = \mathbb{E}_{x \sim p_{data}} \log d(x) + \mathbb{E}_{x \sim p_{model}} \log(1 - d(x)) \tag{10}$$

Это заставляет дискриминатор пытаться обучиться правильно классифицировать примеры как настоящие или поддельные. Одновременно генератор пытается обмануть классификатор, заставив его поверить, что примеры настоящие. В пределе примеры, созданные генератором, неотличимы от настоящих данных, и дискриминатор всегда выводит 1/2. После этого дискриминатор можно выбросить.

Основной побудительный мотив для проектирования ПСС состоит в том, что процесс обучения не нуждается ни в приближенном выводе, ни в аппроксимации градиента статистической суммы. Если $\max_d v(g, d)$ выпукла относительно $\theta^{(g)}$ (как в случае, когда оптимизация производится прямо в пространстве функций плотности вероятности), то процедура гарантированно сходится и асимптотически состоятельна.

К сожалению, на практике обучение ПСС может оказаться трудным, когда g и d представлены нейронными сетями, а функция $\max_d v(g, d)$ не выпуклая. Отсутствие сходимости проблема, которая может привести к недообученности ПСС. В общем случае не гарантируется, что одновременный градиентный спуск по функциям стоимости двух игроков достигнет равновесия. Рассмотрим, к примеру, функцию ценности $v(a, b) = ab$, когда один игрок контролирует a и

несет потери в сумме ab, а второй контролирует b и получает $-ab$. Если мы будем моделировать каждого игрока как совершающего бесконечно малые шаги в направлении градиента, так что каждый игрок уменьшает собственные затраты за счет другого игрока, то a и b выйдут на устойчивую круговую орбиту, а не достигнут точки равновесия в начале координат. Отметим, что точки равновесия в минимаксной игре не являются локальными минимумами v. На самом деле это точки, в которых одновременно достигаются минимумы затрат обоих игроков, т. е. седловые точки v, являющиеся локальными минимумами относительно параметров первого игрока и локальными максимумами относительно параметров второго игрока. Может случиться, что оба игрока по очереди бесконечно увеличивают, а затем уменьшают v, вместо того чтобы оказаться точно в седловой точке, где ни один игрок не может уменьшить своих затрат. Неизвестно, в какой мере эта проблема несходимости затрагивает ПСС.

Также предложена альтернативная формулировка платежей, при которой игра перестает иметь нулевую сумму. При этом ожидаемый градиент такой же, как при обучении с критерием максимального правдоподобия, если только дискриминатор оптимален. Поскольку обучение с критерием максимального правдоподобия сходится, при такой формулировке игры ПСС тоже должна сходиться при наличии достаточного числа примеров. Увы, на практике сходимости не наблюдается, быть может, из-за неоптимальности дискриминатора или высокой дисперсии ожидаемого градиента.

В реалистичных экспериментах наилучшей формулировкой игры ПСС является не игра с нулевой суммой и не эквивалент максимального правдоподобия, введенный с эвристическим обоснованием. Оптимальные результаты получаются, когда генератор стремится увеличить логарифм вероятности, что дискриминатор допустит ошибку, а не уменьшить логарифм вероятности, что дискриминатор сделает правильное предсказание. В обоснование такой формулировки положено одно-единственное наблюдение: при подобной стратегии производная функции стоимости генератора по функции $logit$ дискриминатора остается большой даже в ситуации, когда дискриминатор уверенно отклоняет все примеры генератора.

Стабилизация обучения ПСС остается открытой проблемой. По счастью, обучение ПСС хорошо работает при тщательном выборе архитектуры и гиперпараметров модели. Пример реализации глубокой сверточной ПСС (DCGAN), показывающая отличные результаты в задачах синтеза изображений, и показано, что пространство ее латентного представления улавливает важные

факторы вариативности. На рис. 3 приведены примеры изображений, порожденных генератором DCGAN.

Рис. 3 Изображения, сгенерированные ПСС, обученными на наборе данных LSUN. (Слева) Изображения спален, сгенерированные моделью DCGAN. (Справа) Изображения церквей, сгенерированные моделью LAPGAN.

Задачу обучения ПСС можно упростить, разбив процесс генерации на много уровней детализации. Возможно обучить условные ПСС, которые производят выборку из распределения $p(x|y)$, а не просто из маргинального распределения $p(x)$. Последовательность условных ПСС можно обучить сначала порождать вариант изображения с очень низким разрешением, а затем постепенно добавлять детали. Эта техника называется моделью LAPGAN, поскольку для генерации изображений разного уровня детализации применяется пирамида Лапласа. Генераторы LAPGAN способны обмануть не только дискриминантные сети, но и человека; в экспериментах испытуемые определяли до 40% изображений, сгенерированных сетью, как настоящие данные. На рис. 3 приведены примеры изображений, созданных генератором LAPGAN.

Одна необычная особенность процедуры обучения ПСС заключается в том, что она может аппроксимировать распределения, назначающие нулевую вероятность обучающим примерам. Вместо максимизации логарифма вероятности отдельных точек генераторная сеть обучается очерчивать многообразие, точки которого чем-то напоминают обучающие точки. Парадоксально, но это означает, что модель может присвоить логарифму вероятности тестового набора значение минус бесконечность и тем не менее представлять многообразие, которое, на взгляд человека-наблюдателя, улавливает суть задачи генерации. Это нельзя однозначно назвать ни плюсом, ни минусом. Кроме того, если мы хотим гарантировать, что генераторная сеть назначает

ненулевую вероятность всем точкам, достаточно сделать так, чтобы ее последний слой прибавлял гауссов шум ко всем сгенерированным значениям. Генераторные сети, ведущие себя таким образом, производят выборку из того же распределения, которое получается, когда генераторная сеть используется для параметризации среднего значения условного нормального распределения.

Прореживание, похоже, играет важную роль в дискриминантной сети. В частности, блоки следует стохастически прореживать в процессе вычисления градиента, в направлении которого должна следовать генераторная сеть. Следование градиенту, вычисленному детерминированной версией дискриминатору с весами, поделенными на два, похоже, менее эффективно. К тому же прореживание, кажется, никогда не приносит ничего плохого.

Хотя инфраструктура ПСС предназначена для дифференцируемых генераторных сетей, похожие принципы можно использовать и для обучения моделей других видов. Например, метод самоконтролируемого усиления применяется для обучения генераторной ОМБ (ограниченной машины Больцмана), обманывающей дискриминатор на основе логистической регрессии.

Порождающие сети с сопоставлением моментов

Порождающая сеть с сопоставлением моментов – еще один вид порождающей модели, основанной на дифференцируемых генераторных сетях. В отличие от VAE (вариационный автокодировщик) и ПСС, здесь не нужно комбинировать генераторную сеть с какой-то другой – ни с сетью вывода, как в VAE, ни с дискриминантной сетью, как в ПСС.

Порождающая сеть с сопоставлением моментов обучается методом сопоставления моментов. Его основная идея – обучить генератор так, чтобы многие статистики выборки из модели были максимально похожи на соответствующие статистики выборки из обучающего набора. В этом контексте моментами называются математические ожидания различных степеней случайной величины. Например, первый момент – это среднее, второй – сумма квадратов и т. д. В многомерном случае каждый элемент случайного вектора может быть возведен в разные степени, поэтому моментом может быть любая величина вида

$$\mathbb{E}_x \prod_i x_i^{n_i}, \tag{11}$$

где $n = [n_1, n_2, \dots, n_d]^\top$ – вектор неотрицательных целых чисел.

На первый взгляд кажется, что этот подход вычислительно неразрешим. Например, чтобы сопоставить все моменты вида $x_i x_j$, понадобится минимизировать разность между величинами, количество которых квадратично зависит от размерности x. Более того, даже сопоставления всех первых и вторых моментов будет достаточно только для аппроксимации многомерного

нормального распределения, улавливающего лишь линейные связи между значениями. А наши амбиции простираются на нейронные сети, которые должны улавливать сложные нелинейные связи, так что моментов потребуется куда больше. В ПСС проблемы полного перечисления всех моментов удается избежать благодаря использованию обновляемого дискриминатора, который автоматически концентрирует внимание на той статистике, которую генераторная сеть повторяет наименее эффективно.

Но вместо этого порождающую сеть с сопоставлением моментом можно обучить путем минимизации функции стоимости, называемой максимальным средним расхождением (maximum mean discrepancy – MMD). Эта функция измеряет ошибку на первых моментах в бесконечномерном пространстве, используя неявное отображение в пространство признаков, определяемое некоторой ядерной функцией, в результате чего вычисления с бесконечномерными векторами становятся реальными. Стоимость MMD (максимального среднего расхождения) равна нулю тогда и только тогда, когда два сравниваемых распределения совпадают.

Визуально примеры, выбранные из порождающей сети с сопоставлением моментов, разочаровывают. Но, к счастью, их можно улучшить, скомбинировав генераторную сеть с автокодировщиком. Сначала автокодировщик обучается реконструировать обучающий набор, а затем его кодировщик используется для преобразования всего обучающего набора в кодовое пространство. После этого генераторная сеть обучается генерировать примеры кодов, которые можно отобразить на визуально приемлемые примеры с помощью декодера.

В отличие от ПСС, функция стоимости определена только по отношению к пакету примеров, взятых одновременно из обучающего набора и генераторной сети. Невозможно произвести обновление в виде функции только одного обучающего примера или только одного примера из генераторной сети, поскольку моменты вычисляются как эмпирическое среднее по большому числу примеров. Если размер пакета слишком мал, то MMD (максимальное среднее расхождение) может дать заниженную оценку истинного различия распределений, из которых произведена выборка. Для полного устранения этой проблемы пакета конечного размера вообще недостаточно, но чем больше пакет, тем меньше занижение оценки. Если размер пакета слишком велик, то процедура обучения становится неприемлемо медленной, т. к. для вычисления одного малого шага градиента нужно обработать много примеров.

Как и в случае ПСС, обучить генераторную сеть с помощью MMD можно даже тогда, когда она назначает нулевую вероятность обучающим примерам.

Сверточные порождающие сети

При порождении изображений часто бывает полезно использовать генераторную сеть, включающую сверточную структуру. Для этого применяется «транспонированный» оператор свертки. Этот подход часто дает более реалистичные изображения да к тому же с меньшим числом параметров, чем при использовании полносвязных слоев без разделения параметров.

В сверточных сетях для решения задач распознавания поток информации течет от изображения к верхнему слою сети, где производится обобщение, часто выражаемое меткой класса. По мере распространения вверх по сети информация отбрасывается, т.к. представление изображения становится все более инвариантным к мелким преобразованиям. В генераторной сети все происходит наоборот. По мере того как изображение распространяется по сети, к нему добавляется все больше деталей, и в конечном итоге получается полноценное изображение, где присутствуют все объекты в требуемых положениях, с правильными текстурами и освещением. Основной механизм отбрасывания информации в сверточной сети распознавания – слой пулинга. Но генераторная сеть должна добавлять информацию. Мы не можем включить в генераторную сеть слой инверсного пулинга, поскольку функции пулинга в большинстве своем необратимы. Есть более простая операция – увеличить пространственный размер представления. Применение «антипулинг», который, похоже, дает удовлетворительные результаты. Этот слой соответствует обращению операции max-пулинга при некоторых упрощающих условиях. Во-первых, шаг операции max-пулинга должен совпадать с шириной области пулинга. Во-вторых, предполагается, что максимальный элемент в каждой области пулинга расположен в левом верхнем углу. Наконец, предполагается, что все немаксимальные элементы в каждой области пулинга равны нулю. Это очень сильные и нереалистичные предположения, но они позволяют обратить операцию max-пулинга. Операция антипулинга выделяет память для нулевого тензора, а затем копирует каждое входное значение с пространственной координатой i в выходное значение с пространственной координатой $i \times k$. Целое число k определяет размер области пулинга. Хотя предположения, на которых базируется оператор антипулинга, нереалистичны, последующие слои могут обучиться компенсировать необычный выход, поэтому примеры, генерируемые моделью в целом, визуально приятны.

Авторегрессивные сети

Авторегрессивные сети – это ориентированные вероятностные модели без латентных случайных переменных. Условные распределения вероятности в этих

моделях представлены нейронными сетями (иногда очень простыми типа логистической регрессии). Модели соответствует полный граф. Совместное распределение наблюдаемых переменных в таких моделях с помощью цепного правила вероятностей разлагается в произведение условных распределений вида $P(x_d|x_{d-1}, \dots, x_1)$. Такие модели под названием «полностью видимые байесовские сети» (ПВБС, англ. FVBN) успешно использовались в различных формах, сначала с логистической регрессией в качестве каждого условного распределения, а затем с нейронными сетями со скрытыми блоками. В некоторых вариантах авторегрессивных сетей, например NADE, которую мы еще рассмотрим, можно реализовать некий вид разделения параметров, что дает как статистический (меньше уникальных параметров), так и вычислительный (меньше объем вычислений) выигрыш. Это еще один пример повторяющегося в глубоком обучении мотива повторного использования признаков.

Линейные авторегрессивные сети

В простейшей форме авторегрессивных сетей нет ни скрытых блоков, ни разделения параметров. Каждое условное распределение $P(x_i|x_{i-1}, \dots, x_1)$ параметризуется как линейная модель (линейная регрессия в случае вещественных данных, логистическая регрессия в случае бинарных данных и softmax-регрессия в случае дискретных данных). Эта модель впервые была предложена в 1998 году, в ней $O(d^2)$ параметров, где d – количество переменных в модели. Она показана на рис. 4.

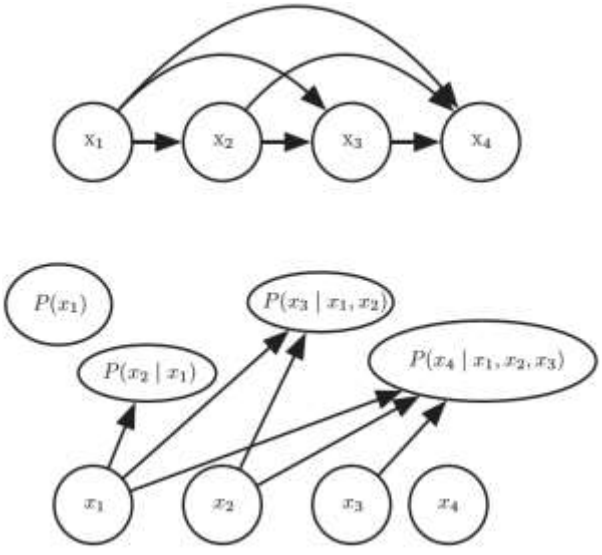

Рис. 4 Полностью видимая байесовская сеть предсказывает i-ю переменную по $i-1$ предыдущим. (Сверху) Ориентированная графическая модель ПВБС (полностью видимые байесовские сети) (Снизу). Граф вычислений для логистической ПВБС, в которой каждое предсказание делается линейным предиктором

Если переменные непрерывны, то линейная авторегрессивная сеть – просто еще одна формулировка многомерного нормального распределения, улавливающего линейные попарные взаимодействия между наблюдаемыми величинами.

По сути своей линейные авторегрессивные сети – это обобщение методов линейной классификации на порождающее моделирование. Поэтому у них такие же плюсы и минусы, как у линейных классификаторов. Их точно так же можно обучать с помощью выпуклых функций потерь, и иногда они допускают решение в замкнутой форме (как в случае нормального распределения). Как и линейный классификатор, сама по себе модель не предлагает никакого способа увеличения емкости, так что для этой цели следует использовать такие приемы, как расширение базиса входа или трюк с ядром.

Нейронные авторегрессивные сети

Нейронным авторегрессивным сетям соответствует такая же направленная слева направо графическая модель, как логистическим авторегрессивным сетям (рис. 4), но внутри этой графической структуры используется другая параметризация условных распределений. Она мощнее в том смысле, что емкость модели можно увеличивать как угодно и, значит, аппроксимировать любое совместное распределение. Кроме того, новая параметризация улучшает обобщаемость благодаря принципу разделения параметров и признаков, присущему глубокому обучению вообще. Модель была предложена, чтобы уйти от проклятия размерности, имеющего место в традиционных табличных графических моделях с такой же структурой, как на рис. 4. В табличных дискретных вероятностных моделях каждое условное распределение представлено таблицей вероятностей, в которой каждой возможной конфигурации участвующих переменных соответствуют одна ячейка и один параметр. Использование вместо этого нейронной сети дает два преимущества:

1) параметризация каждого $P(x_i|x_{i-1}, \dots, x_1)$ нейронной сетью с $(i-1) \times k$ входами и k выходами (если переменные дискретны и принимают k значений, представленных унитарным кодом) позволяет оценить условную вероятность, не требуя экспоненциального числа параметров (и примеров), но при этом еще и способна уловить зависимости высшего порядка между случайными величинами;

2) вместо отдельной нейронной сети для предсказания каждого x_i направленные слева направо связи (рис. 5) позволяют объединить все сети в одну. Иначе говоря, это означает, что признаки скрытого слоя, вычисленные для предсказания x_i, можно повторно использовать для предсказания $x_{i+k}(k > 0)$. Таким образом, скрытые блоки организованы в группы, обладающие тем

свойством, что все блоки i-й группы зависят только от входных значений $x_1, \dots, x_i$. Параметры, используемые для вычисления этих скрытых блоков, совместно оптимизируются с целью улучшить предсказание всех переменных в последовательности. Это пример принципа повторного использования, который пронизывает все глубокое обучения – от архитектур рекуррентных и сверточных сетей до многозадачного обучения и переноса обучения.

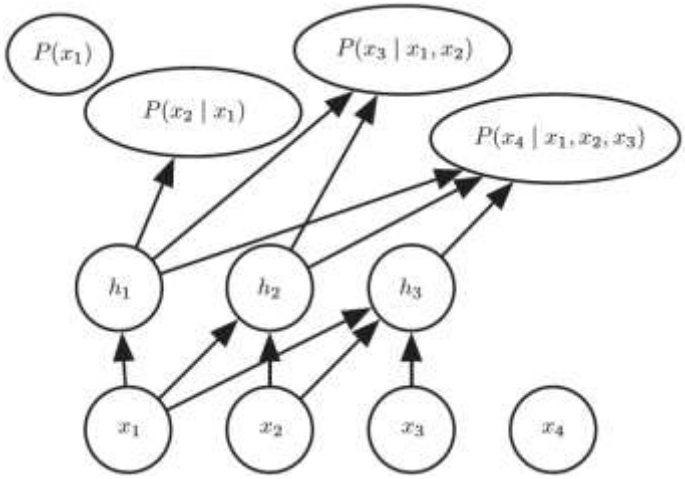

Рис. 5. Нейронная авторегрессивная сеть предсказывает i-ю переменную x_i по $i-1$ предыдущим, но параметризована так, что признаки (группы скрытых блоков, обозначенные h_i), являющиеся функциями от $x_1, \dots, x_i$, можно повторно использовать для предсказания всех последующих переменных $x_{i+1}, x_{i+2}, \dots, x_d$

Каждое $P(x_i|x_{i-1}, \dots, x_1)$ может представлять условное распределение, если выходы нейронной сети будут предсказывать параметры условного распределения x_i. Хотя первоначально авторегрессивные сети работали с чисто дискретными многомерными данными (с сигмоидным выходным блоком для случайных величин с распределением Бернулли или softmax-блоком для величин с категориальным распределением), они естественно обобщаются на непрерывные величины или совместные распределения, включающие как дискретные, так и непрерывные величины.

NADE – Нейронный авторегрессивный оцениватель плотности

Нейронный авторегрессивный оцениватель плотности (neural auto-regressive density estimator – NADE) – недавно появившаяся и уже ставшая очень успешной форма нейронной авторегрессивной сети. Связность в ней такая же, как в оригинальной нейронной авторегрессивной сети, но введена дополнительная схема разделения параметров, показанная на рис. 6. Параметры скрытых блоков из разных групп j разделяются.

Веса $W'_{j,k,i}$ связей между i-м входом x_i и k-ым элементом j-ой группы скрытых блоков $h_k^{(j)} (j \geq i)$ разделяются между группами:

$$W'_{j,k,i} = W_{k,i}. \quad (12)$$

Остальные веса (для $j < i$) равны нулю.

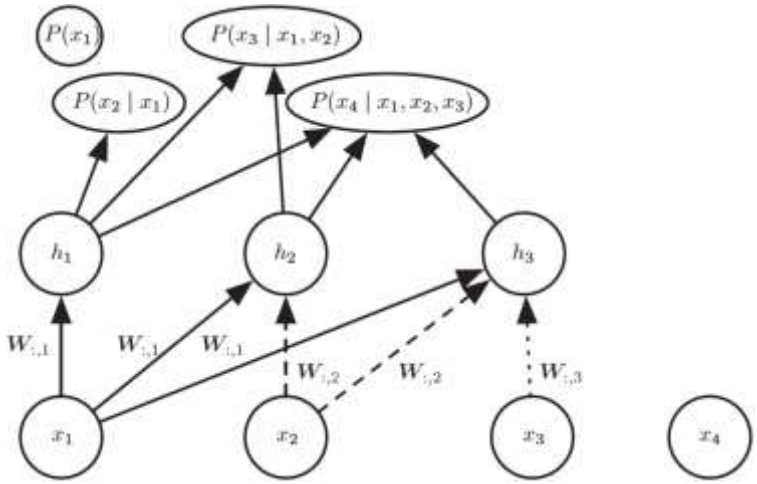

Рис. 6 Нейронный авторегрессивный оцениватель плотности (NADE). Скрытые блоки организованы в группы $h^{(j)}$, так что только входы $x_1, \ldots, x_i$ участвуют в вычислении $h^{(i)}$ и предсказании $P(x_j|x_{j-1}, \ldots, x_1)$ для $j > i$. NADE отличается от более ранних нейронных авторегрессивных сетей использованием специальной схемы разделения весов: значение $W'_{j,k,i} = W_{k,i}$ одинаково (на рисунке это обозначено одинаковым стилем линий для всех экземпляров повторяющегося веса) для всех весов связей, идущих из x_i в k-ый блок любой группы $j \geq i$. Напомним, что вектор $(W_{1,i}, W_{2,i}, \ldots, W_{n,i})$ обозначается $W_{:,i}$

Выбор этой схемы разделения, где прямое распространение в NADE (нейронный авторегрессивный оцениватель плотности) чем-то напоминает вычисления, которые производятся в выводе среднего поля для восполнения отсутствующих значений в ОМБ (ограниченной машине Больцмана). Этот вывод среднего поля соответствует выполнению рекуррентной сети с разделяемыми весами, и первый шаг вывода такой же, как в NADE. Единственное отличие состоит в том, что в NADE веса связей скрытых блоков с выходными параметризуются независимо от весов связей входных блоков со скрытыми. В ОМБ матрица весов связей скрытые–выходные является транспонированной к матрице весов связей входные–скрытые. Архитектуру NADE можно обобщить, так чтобы она имитировала не один, а k временных шагов рекуррентного вывода среднего поля. Этот подход называется NADE-k.

Как уже отмечалось, авторегрессивные сети можно распространить на обработку непрерывных данных. Особенно эффективным и общим способом параметризации непрерывной плотности является смесь гауссовых распределений с весами α_i (коэффициент или априорная вероятность i-й компоненты),

условными средними μ_i и условными дисперсиями σ_i^2. В модели RNADE такая параметризация используется для обобщения NADE ((нейронный авторегрессивный оцениватель плотности)) на вещественные значения. Как и в других сетях со смесовой плотностью, параметры этого распределения являются выходами сети, причем вероятности весов компонент порождаются softmax-блоком, а дисперсии положительны. Метод стохастического градиентного спуска может оказаться численно неустойчивым из-за взаимодействий между условными средними и условными дисперсиями. Для преодоления этой трудности на этапе обратного распространения используется псевдоградиент, заменяющий градиент по среднему.

В еще одном очень интересном обобщении нейронных авторегрессивных архитектур удалось избавиться от необходимости выбирать произвольный порядок наблюдаемых переменных. Идея авторегрессивной сети – обучить сеть справляться с любым порядком за счет того, что порядок выбирается случайно, а скрытым блокам передается информация о том, какие входные данные наблюдались (находятся справа от вертикальной черты в формуле условной вероятности), а какие считаются отсутствующими и подлежат предсказанию (находятся слева от вертикальной черты). Это хорошо, потому что позволяет весьма эффективно использовать обученную авторегрессивную сеть для решения любой проблемы вывода (т.е. предсказывать или производить выборку из распределения вероятности любого подмножества переменных при условии любого другого подмножества). Наконец, поскольку возможно много порядков переменных (n! для n переменных) и каждый порядок o переменных дает различные вероятности $p(x|o)$, мы можем образовать ансамбль моделей для нескольких значений o:

$$p_{ensemble}(x) = \frac{1}{k}\sum_{i=1}^{k} p(x|o^{(i)}) \quad (13)$$

Этот ансамбль моделей обычно лучше обообщается и назначает тестовому набору более высокую вероятность, чем отдельная модель, определенная одним упорядочением.

В той же работе авторы предлагают глубокие варианты этой архитектуры, но, к сожалению, вычисления при этом сразу же становятся такими же дорогостоящими, как в оригинальной нейронной авторегрессивной сети. Первый и выходной слои все еще можно вычислить за $O(nh)$ операций умножения-сложения, как в стандартном алгоритме NADE, где h – число скрытых блоков (размер групп h_i на рис. 6 и 5), тогда как в некоторых работах таких операций $O(n^2h)$. Но для других скрытых слоев сложность вычислений имеет порядок

$O(n^2h^2)$, если каждая «предыдущая» группа в слое l участвует в предсказании «следующей» группы в слое $l+1$, а n – число групп из h скрытых блоков в каждом слое. Если предположить, что i-я группа в слое $l+1$ зависит только от i-й группы в слое l, то сложность уменьшится до $O(nh^2)$, но это все равно в h раз хуже, чем в стандартном NADE.