ЛЕКЦИЯ 6. Другие порождающие сети

Выборка из автокодировщиков

Мы видели, что многие виды автокодировщиков обучаются распределению данных. Существуют тесные связи между сопоставлением рейтингов, шумоподавляющими автокодировщиками и сжимающими автокодировщиками. Они показывают, что некоторые автокодировщики каким-то образом обучаются распределению данных. Но мы еще не видели, как производить выборку из таких моделей.

Некоторые автокодировщики, например вариационные, явно представляют распределение вероятности и допускают прямую предковую выборку. Для большинства других необходима выборка MCMC-методами (методы Монте-Карло по схеме марковских цепей).

Сжимающие автокодировщики предназначены для восстановления оценки касательной плоскости к многообразию данных. Это означает, что повторяющееся кодирование и декодирование с привнесенным шумом индуцирует случайное блуждание по поверхности многообразия. Эта техника диффузии на многообразии является разновидностью марковской цепи.

Существует также более общая марковская цепь, способная производить выборку из любого шумоподавляющего автокодировщика.

Марковская цепь, ассоциированная с произвольным шумоподавляющим автокодировщиком

В обсуждении выше остался открытым вопрос о том, какой шум привносить и где взять марковскую цепь, которая будет генерировать примеры из распределения, оцененного автокодировщиком. Построить такую марковскую цепь для обобщенных шумоподавляющих автокодировщиков. Такие автокодировщики задаются шумоподавляющим распределением для выборки из оценки чистого входа при известном зашумленном входе.

Каждый шаг марковской цепи, генерирующей примеры из оценки распределения, состоит из следующих подшагов, показанных на рис. 1:

1. Начав с предыдущего состояния x, привнести искажающий шум, выбирая $\tilde{x}$ из $C(\tilde{x}|x)$.

2. Закодировать $\tilde{x}$ в $h = f(\tilde{x})$.

3. Декодировать h и получить параметры $\omega = g(h)$ распределения $p(x|\omega = g(h)) = p(x|\tilde{x})$.

4. Выбрать следующее состояние x из $p(x|\omega = g(h)) = p(x|\tilde{x})$.

Если автокодировщик $p(x|\tilde{x})$ дает состоятельную оценку соответствующего условного распределения, то стационарное распределение вышеуказанной

марковской цепи дает состоятельную оценку (хотя и неявную) порождающего данные распределения x.

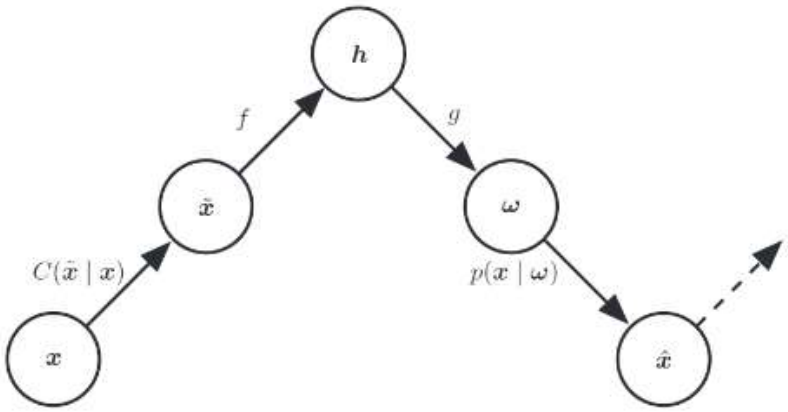

Рис. 1. Шаг марковской цепи, ассоциированной с обученным шумоподавляющим автокодировщиком, которая генерирует примеры из вероятностной модели, неявно обученной с критерием шумоподавляющего логарифмического правдоподобия. Каждый шаг состоит из: (a) привнесения шума в состояние x с помощью искажающего процесса C, который дает $\tilde{x}$; (b) кодирования $\tilde{x}$ с помощью функции f и получения $h = f(\tilde{x})$; (c) декодирования результата с помощью функции g с получением параметров ω для реконструкции распределения и (d) выборки нового состояния из реконструированного распределения $p(x|\omega = g(f(\tilde{x})))$ при известном ω. В типичном случае среднеквадратической ошибки реконструкции $g(h) = \hat{x}$, которая оценивает $\mathbb{E}[x|\tilde{x}]$, искажение сводится к прибавлению гауссова шума, а выборка из $p(x|\omega)$ – к повторному прибавлению гауссова шума к реконструкции $\hat{x}$. Второе прибавление шума должно соответствовать среднеквадратическим ошибкам реконструкции, а привнесенный шум – это гиперпараметр, управляющий скоростью приработки и степенью сглаживания эмпирического распределения оценивателем. В приведенном примере только условные распределения C и p – стохастические шаги (вычисление f и g детерминировано), хотя шум можно привносить и внутри автокодировщика, как в порождающих стохастических сетях.

Возвратная процедура обучения

Возвратная (walk-back) процедура обучения это способ ускорить сходимость порождающего обучения шумоподавляющих автокодировщиков. Вместо выполнения одного шага реконструкции кодирование-декодирование в этой процедуре попеременно выполняется несколько стохастических шагов кодирования-декодирования (как в порождающей марковской цепи). Процедура

начинается с обучающего примера (как в алгоритме сопоставительного расхождения, описанном в разделе 18.2) и штрафует последние вероятностные реконструкции (или все реконструкции на пути).

Обучение с k шагами эквивалентно (в том смысле, что достигается то же самое стационарное распределение) обучению с одним шагом, но обладает практическим преимуществом: паразитные моды вдали от данных устраняются эффективнее.

Рис. 2 Фиксация правой половины изображения и выполнение марковской цепи путем повторной выборки только из левой половины на каждом шаге. Примеры взяты из порождающей стохастической сети, обученной реконструировать цифры из набора MNIST на каждом шаге с помощью возвратной процедуры

Порождающие стохастические сети

Порождающие стохастические сети (generative stochastic networks – GSN) – это обобщение шумоподавляющих автокодировщиков. Они включают латентные переменные h в порождающей марковской цепи, помимо видимых переменных (обычно обозначаемых x).

GSN параметризуется двумя условными распределениями вероятности, описывающими один шаг марковской цепи.

1. $p(x^{(k)}|h^{(k)})$ говорит, как сгенерировать следующую видимую переменную, если известно текущее латентное состояние. Такое «распределение реконструкции» встречается также в шумоподавляющих автокодировщиках, ОМБ (ограниченные машины Больцмана), ГСД (глубокие сети доверия) и ГМБ (глубокие машины Больцмана).

2. $p(h^{(k)}|h^{(k-1)}, x^{(k-1)})$ говорит, как обновить латентную переменную состояния, если известны предыдущее латентное состояние и видимая переменная.

Шумоподавляющие автокодировщики и GSN (порождающие стохастические сети) отличаются от классических вероятностных моделей (ориентированных и неориентированных) тем, что параметризуют сам порождающий процесс, а не математическую спецификацию совместного распределения видимых и латентных переменных. Последнее же (если существует) определяется неявно как стационарное распределение порождающей марковской цепи.

Условия существования стационарного распределения довольно мягкие – такие же, как в стандартных MCMC-методах (методы Монте-Карло по схеме марковских цепей). Это необходимые условия приработки цепи, но существуют такие переходные распределения (например, детерминированные), для которых они нарушаются.

Можно представить себе различные критерии обучения GSN (порождающие стохастические сети). Предложено просто использовать логарифм вероятности реконструкции для видимых блоков, как в шумоподавляющих автокодировщиках. Для этого нужно зафиксировать $x^{(0)} = x$, наблюдаемому примеру, и максимизировать вероятность порождения x на протяжении какого-то количества последующих временных шагов, т. е. максимизировать $\log p(x^{(k)} = x|h^{(k)})$, где $h^{(k)}$ выбирается из цепи при условии $x^{(0)} = x$. Чтобы оценить градиент $\log p(x^{(k)} = x|h^{(k)})$ по другим частям модели, используется трюк с перепараметризацией, а для улучшения сходимости – возвратная процедура обучения.

Дискриминантные GSN (порождающие стохастические сети)

В описанных выше методах для порождения примеров использовалась либо MCMC-выборка, либо предковая выборка, либо та или иная их комбинация. Хотя это наиболее популярные подходы к порождающему моделированию, они ни в коем случае не единственные.

Для обучения порождающей модели разработана схема инверсии диффузии, основанная на неравновесной термодинамике. Основная идея состоит в том, что у распределений вероятности, из которых мы хотим производить выборку, имеется структура. Эта структура может постепенно разрушаться процессом диффузии, который изменяет распределение в сторону большей энтропии.

Для формирования порождающей модели мы можем обратить этот процесс, обучив модель, которая постепенно восстанавливает структуру бесструктурного распределения. Итеративно применяя процесс сближения распределения с целевым, мы можем подойти к целевому распределению. Этот подход напоминает MCMC-методы (методы Монте-Карло по схеме марковских цепей) в том смысле,

что для порождения выборки нужно много итераций. Однако модель определена как распределение вероятности, порождаемое последним шагом цепи. В этом смысле итеративная процедура не индуцирует никакой аппроксимации. Описанный подход также очень близок к порождающей интерпретации шумоподавляющего автокодировщика. Как и шумоподавляющий автокодировщик, инверсия диффузии обучает оператор перехода, который пытается вероятностно компенсировать эффект сложения с каким-то шумом. Разница в том, что инверсия диффузии требует отменить только один шаг диффузионного процесса, а не пройти назад по всему пути к чистым данным. Тем самым устраняется дилемма, присущая обычной для шумоподавляющих автокодировщиков целевой функции логарифма вероятности реконструкции: при малом уровне шума обучаемый видит только конфигурации рядом с примерами данных, а при большом задача становится почти неразрешимой (поскольку шумоподавляющее распределение становится слишком сложным и многомодальным). В случае же целевой функции инверсии диффузии обучаемый может более точно восстановить форму функции плотности в окрестности данных, а заодно избавиться от паразитных мод вдали от данных.

Еще один подход к генерации примеров – приближенные байесовские вычисления (approximate Bayesian computation – ABC). В этом случае примеры отклоняются или модифицируются, так чтобы моменты выбранных функций примеров совпадали с моментами желаемого распределения. В этой идее используются моменты примеров, как в алгоритме сопоставления моментов, но есть и различия, поскольку здесь производится модификация самих примеров, а не обучение модели автоматически выдавать примеры с правильными моментами.

Идеи ABC можно использовать в контексте глубокого обучения для формирования траекторий MCMC в порождающих стохастических сетях.

Оценивание порождающих моделей

Исследователям, изучающим порождающие модели, часто бывает необходимо сравнить две модели, обычно чтобы продемонстрировать, что новая модель лучше улавливает некоторое распределение, чем предыдущие.

Это может оказаться непростой задачей. Нередко точно вычислить логарифм вероятности данных в модели невозможно, приходится довольствоваться только аппроксимацией. В таких случаях важно отчетливо понимать и сообщать аудитории, что именно измеряется. Предположим, к примеру, что мы вычисляем стохастическую оценку логарифмического правдоподобия модели A и детерминированную нижнюю границу логарифмического правдоподобия модели B. Если модель A получила больше

баллов, чем модель B, то какая из них лучше? Если нас интересует, какая модель дает лучшее внутреннее представление распределения, то ответить на этот вопрос нельзя, если только нет какого-то способа узнать, насколько точна нижняя граница для модели B. Если же нам интересно практическое использование модели, например для обнаружения аномалий, то будет справедливо судить модели на основе критерия, относящегося к конкретной задаче, например по результатам ранжирования тестовых примеров с помощью таких критериев, как точность и полнота.

Еще одна тонкость оценивания порождающих моделей состоит в том, что выработка критериев оценки сама по себе представляет трудную научную задачу. Может оказаться очень сложно установить, что сравнение моделей производится справедливо. Предположим, к примеру, что мы используем метод AIS (annealed importance sampling) стратегия выборки по значимости с отжигом для получения оценки $\log Z$ с целью вычислить $\log \tilde{p}(x) - logZ$ для новой придуманной нами модели. Вычислительно экономная реализация AIS (стратегия выборки по значимости с отжигом) может не найти несколько мод модельного распределения и дать заниженную оценку Z, что приведет к завышенной оценке $\log p(x)$. Таким образом, трудно сказать, что стало причиной высокой оценки правдоподобия: хорошая модель или плохая реализация AIS.

В других разделах машинного обучения обычно допускается некоторая вариативность на этапе предобработки данных. Например, при сравнении верности алгоритмов распознавания объектов обычно разрешается производить предобработку входных изображений немного по-разному в соответствии с требованиями, предъявляемыми каждым алгоритмом. Порождающее моделирование устроено иначе – любые изменения в способе предобработки, пусть даже совсем незначительные и незаметные, абсолютно недопустимы. Всякое изменение входных данных изменяет подлежащее выявлению распределение и кардинальным образом меняет задачу. Например, умножение входных данных на 0.1 искусственно повышает правдоподобие в 10 раз.

Проблемы предобработки часто возникают при проверке порождающих моделей на эталонном наборе данных MNIST, одном из самых популярных для тестирования таких моделей. В этом наборе есть только полутоновые изображения. В одних моделях изображения из MNIST рассматриваются как точки в вещественном векторном пространстве, в других – как бинарные изображения. А в третьих полутоновые значения яркости трактуются как вероятности бинарных примеров. Важно сравнивать вещественные модели только с другими вещественными моделями, а бинарные – только с другими бинарными.

В противном случае правдоподобие будет измеряться в разных пространствах. Для бинарных моделей логарифмическое правдоподобие не может быть больше нуля, тогда как в вещественных оно не ограничено сверху, будучи результатом измерения плотности. При сравнении бинарных моделей важно, чтобы применялся один и тот же вид бинаризации. Например, для сопоставления полутоновому пикселю значения 0 или 1 мы можем сравнить его с порогом 0.5 или произвести случайную выборку, в которой вероятность получить 1 определяется яркостью пикселя. Если используется случайная бинаризация, то мы можем бинаризовать весь набор данных сразу или выбрать разные случайные примеры для каждого шага обучения, а затем произвести множественную выборку для оценивания. Все три схемы дадут совершенно разные значения правдоподобия, а при сравнении разных моделей важно, чтобы использовалась одна и та же схема бинаризации для обучения и оценивания. На самом деле при выполнении единственного шага случайной бинаризации обычно создается общий файл, содержащий ее результаты, чтобы исключить расхождения из-за различных исходов шага бинаризации.

Поскольку способность порождать реалистичные примеры из распределения данных – одна из целей порождающей модели, на практике такие модели часто оценивают, визуально исследуя примеры. Лучше, когда это делает не сам исследователь, а участник эксперимента, которому неизвестен источник происхождения примеров. К сожалению, бывает так, что очень плохая вероятностная модель порождает очень хорошие примеры. Общепринятый способ проверить, что модель не просто копирует какие-то обучающие примеры. Идея в том, чтобы для некоторых порожденных примеров показать их ближайших соседей в обучающем наборе согласно евклидову расстоянию в пространстве x. Эта проверка направлена на то, чтобы выявить случай, когда модель переобучена и просто воспроизводит обучающие примеры. Может даже случиться, что модель одновременно переобучена и недообучена и тем не менее порождает примеры, которые по отдельности выглядят отлично. Представьте себе порождающую модель, обученную на изображениях собак и кошек, которая просто научилась воспроизводить изображения собак. Очевидно, что такая модель переобучена, поскольку она не порождает изображения, которых не было в обучающем наборе, но она также недообучена, т. к. назначает нулевую вероятность обучающим изображениям кошек. Тем не менее человек сочтет, что каждое отдельное изображение собаки высокого качества. Это простой пример – наблюдатель, просмотревший много примеров, заметит отсутствие кошек. В более реалистичных условиях порождающая модель, обученная на данных с десятками

тысяч мод, может проигнорировать небольшое число мод, и человеку будет нелегко заметить, что какая-то вариация отсутствует.

Поскольку визуальное качество примеров – ненадежный путеводитель, мы часто оцениваем также логарифмическое правдоподобие, которое модель назначает данным, если это вычислительно осуществимо. К сожалению, в некоторых случаях правдоподобие не измеряет интересующих нас атрибутов модели. Например, на наборе данных MNIST вещественная модель может получить произвольно высокое правдоподобие, если назначит произвольно низкую дисперсию пикселям фона, которые никогда не изменяются. Модели и алгоритмы, которые обнаруживают такие постоянные признаки, могут быть вознаграждены не по заслугам, потому что особой пользы в этом свойстве нет. Потенциальная возможность достичь стоимости, стремящейся к минус бесконечности, существует для любого вида задач с критерием максимального правдоподобия с вещественными значениями, но особенно от этого страдают порождающие модели, оцениваемые на наборе MNIST, потому что количество тривиально предсказываемых выходных значений очень велико. Поэтому возникает настоятельная необходимость в разработке других способов оценивания порождающих моделей.

Приведен обзор многих проблем, возникающих при оценивании порождающих моделей, включающий и описанные выше соображения. Авторы подчеркивают, что порождающие модели применяются для самых разных целей и что выбор метрики должен соответствовать назначению модели. Так, одни порождающие модели лучше назначают высокую вероятность самым реалистичным точкам, тогда как другие преуспевают в редком назначении высокой вероятности нереалистичным точкам. Такие различия могут быть связаны с тем, проектировалась ли модель для минимизации $D_{KL}(p_{data}||p_{model})$ или $D_{KL}(p_{model}||\,p_{data})$. К сожалению, даже если ограничиться использованием только метрик, отвечающих задаче, у всех известных в настоящее время метрик имеются серьезные недостатки. Поэтому одно из самых важных направлений исследований в области порождающего моделирования – не улучшение самих моделей, а проектирование новых методов измерения успеха.

Обучение порождающих моделей со скрытыми блоками – эффективный способ научить модель понимать мир, представленный обучающими данными. Обучившись распределению $p_{model}(x)$ и представлению $p_{model}(h|x)$, порождающая модель может давать ответы на многие вопросы о связях между входными переменными в x и предлагать другие способы представления x путем вычисления математических ожиданий h на разных слоях иерархии.

Порождающие модели выполняют обещание снабдить системы ИИ инфраструктурой для понимания многообразных интуитивных концепций и наделить их возможностью рассуждать об этих концепциях в условиях неопределенности.