ЛЕКЦИЯ 1. Автокодировщики

Автокодировщиком называется нейронная сеть, обученная пытаться скопировать свой вход в выход. На внутреннем уровне в ней имеется скрытый слой h, который описывает код, используемый для представления входа. Можно считать, что сеть состоит из двух частей: функция кодирования $h = f(x)$ и декодер, порождающий реконструкцию $r = g(h)$. Эта архитектура показана на рис. 1.1. Если автокодировщику удается просто обучиться всюду сохранять тождество $g(f(x)) = x$, то это не особенно полезно. На самом деле автокодировщики проектируются так, чтобы они не могли обучиться идеальному копированию. Обычно налагаются ограничения, позволяющие копировать только приближенно и только тот вход, который похож на обучающие данные. Поскольку модель заставляют расставлять подлежащие копированию аспекты данных по приоритетам, то часто она обучается полезным свойствам данных.

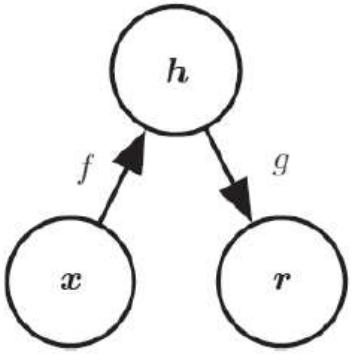

Рис. 1.1 Общая структура автокодировщика, отображающего вход x на выход r (называемый реконструкцией) через внутреннее представление, или код h. Автокодировщик состоит из двух частей: кодировщик f (отображение x в h) и декодер g (отображение h в r)

В современных автокодировщиках идея кодировщика и декодера обобщена с детерминированных на стохастические отображения $p_{encoder}(h|x)$ и $p_{decoder}(x|h)$.

Идея автокодировщиков исторически является частью ландшафта нейронных сетей уже несколько десятилетий. Традиционно их использовали для понижения размерности и обучения признакам. Выявленные недавно теоретические связи между автокодировщиками и моделями с латентными переменными вывели автокодировщики на передний фронт порождающего моделирования. Автокодировщики можно рассматривать как частный случай сетей прямого распространения и обучать, применяя те же методы, обычно мини-пакетный градиентный спуск в направлении градиентов, вычисленных в ходе обратного распространения. Но, в отличие от общих сетей прямого

распространения, автокодировщики можно обучать также с помощью рециркуляции – алгоритма обучения, основанного на сравнении активаций сети на оригинальных и реконструированных входных данных.

Рециркуляция считается биологически более правдоподобным механизмом, чем обратное распространение, но редко применяется в приложениях машинного обучения.

1.1. Понижающие автокодировщики

Копирование входа в выход может показаться бесполезным делом, но, как правило, нас не интересует выход декодера. Вместо этого мы надеемся, что, обучая автокодировщик задаче копирования входа, мы получим код h, обладающий полезными свойствами. Один из способов получить такие свойства – наложить ограничение, согласно которому размерность h должна быть меньше размерности x. Такие автокодировщики называются понижающими (undercomplete). Обучение понижающего представления заставляет автокодировщик улавливать наиболее отличительные признаки обучающих данных.

Процесс обучения описывается просто как минимизация функции потерь

$$L(x, g(f(x))), \tag{1.1}$$

где функция L штрафует $g(f(x))$ за непохожесть на x; например, это может быть среднеквадратическая ошибка.

Если декодер – линейная функция, а L – среднеквадратическая ошибка, то понижающий автокодировщик обучается тому же подпространству, что метод главных компонент (principal components analysis – PCA). В таком случае автокодировщик, обученный выполнять задачу копирования, в качестве побочного эффекта находит главное подпространство обучающих данных.

Следовательно, автокодировщики с нелинейными функциями кодирования f и декодирования g могут обучиться более мощным нелинейным обобщениям PCA. К сожалению, если разрешить слишком большую емкость кодировщика и декодера, то автокодировщик будет решать задачу копирования, не извлекая полезной информации о распределении данных. Теоретически можно вообразить автокодировщик с одномерным кодом, но очень мощным нелинейным кодировщиком, который обучится представлять каждый обучающий пример $x^{(i)}$ кодом i. Декодер можно было бы обучить отображению этих целочисленных индексов обратно на значения конкретных обучающих примеров. Такой сценарий, на практике не встречающийся, ясно показывает, что автокодировщик, обученный копированию, может не собрать никакой полезной информации о наборе данных, если его емкость слишком велика.

1.2. Регуляризированные автокодировщики

Понижающие автокодировщики, для которых размерность кода меньше размерности входа, могут обучиться наиболее отличительным признакам распределения данных. Мы поняли, что такие автокодировщики не обучаются ничему полезному, если емкость кодировщика и декодера слишком велика.

Похожая проблема возникает, если скрытый код имеет такую размерность, как вход, а также в случае повышающего автокодировщика, когда размерность скрытого кода больше размерности входа. В этих случаях даже линейные кодировщик и декодер могут научиться копировать вход в выход, не узнав ничего полезного о распределении данных.

В идеале можно было бы успешно обучить любую архитектуру автокодировщика, выбрав размерность кода и емкость кодировщика и декодера, исходя из сложности моделируемого распределения. Такую возможность предоставляют регуляризированные автокодировщики. Вместо того чтобы ограничивать емкость модели, стремясь сделать кодировщик и декодер мелкими, а размер кода малым, регуляризированный автокодировщик использует такую функцию потерь, которая побуждает модель к приобретению дополнительных свойств, помимо способности копировать вход в выход. К числу таких свойств относятся разреженность представления, малая величина производной представления и устойчивость к шуму или отсутствию части входных данных. Регуляризированный автокодировщик может быть нелинейным и повышающим и тем не менее узнать что-то полезное о распределении данных, даже если емкость модели достаточно велика, чтобы обучить тривиальную тождественную функцию.

Помимо описанных ниже методов, которые наиболее естественно интерпретируются как регуляризированные автокодировщики, почти любая порождающая модель с латентными переменными и процедурой вывода (для вычисления латентных представлений по входу) может рассматриваться как некий вид автокодировщика. Существуют два подхода к порождающему моделированию, подчеркивающие такую связь с автокодировщиками: ведущие происхождение от машины Гельмгольца, например вариационный автокодировщик, и порождающие стохастические сети. Эти модели естественно обучаются повышающему кодированию входа с высокой емкостью и оказываются полезными даже без регуляризации кодирования. Созданные ими коды полезны, потому что модель обучалась приближенно максимизировать вероятность обучающих данных, а не копировать вход в выход.

1.2.1. Разреженные автокодировщики

Разреженным называется автокодировщик, для которого критерий обучения включает штраф разреженности $\Omega(h)$ на кодовом слое h в дополнение к ошибке реконструкции:

$$L(x, g(f(x))) + \Omega(h), \tag{1.2}$$

где $g(h)$ – выход декодера и обычно $h = f(x)$ – выход кодировщика.

Разреженные автокодировщики чаще всего применяются для обучения признакам в интересах другой задачи, например классификации. Автокодировщик, регуляризированный с целью добиться разреженности, должен откликаться на уникальные статистические особенности набора данных, на котором обучался, а не просто выступать в роли тождественной функции. Поэтому обучение копированию со штрафом разреженности может дать модель, которая попутно обучилась каким-то полезным признакам.

Штраф $\Omega(h)$ можно рассматривать как регуляризирующий член, добавленный к сети прямого распространения, основная задача которой – копировать вход в выход (целевая функция для обучения без учителя) и, возможно, решать еще какую-то задачу обучения с учителем, зависящую от найденных разреженных признаков.

В отличие от других регуляризаторов, например снижения весов, у этого регуляризатора нет очевидной байесовской интерпретации. Обучение со снижением весов и другими регуляризирующими штрафами можно интерпретировать как аппроксимацию байесовского вывода максимумом апостериорной вероятности с добавлением регуляризирующего штрафа, который соответствует априорному распределению вероятности параметров модели. С этой точки зрения, регуляризированное максимальное правдоподобие соответствует максимизации $p(\theta|x)$, что эквивалентно максимизации $\log p(x|\theta) + \log p(\theta)$. Член $\log p(x|\theta)$ – обычное логарифмическое правдоподобие данных, а член $\log p(\theta)$, логарифмическое априорное распределение параметров, знаменует предпочтение определенным значениям θ. Для регуляризированных автокодировщиков такая интерпретация не годится, потому что регуляризатор зависит от данных и поэтому по определению не может считаться априорным распределением в формальном смысле слова. Однако мы по-прежнему можем считать, что регуляризирующие члены неявно выражают предпочтение некоторым функциям.

Вместо того чтобы считать штраф разреженности регуляризатором для копирования данных, мы можем рассматривать всю инфраструктуру разреженного автокодирования как обучение порождающей модели с латентными

переменными, аппроксимирующее максимальное правдоподобие. Предположим, что имеется модель с видимыми переменными x, латентными переменными h и явным совместным распределением $p_{model}(x,h) = p_{model}(h)p_{model}(x|h)$. Мы называем $p_{model}(h)$ априорным распределением латентных переменных и считаем, что оно представляет априорную веру модели в то, что она увидит x. Эта трактовка отличается от предыдущего употребления слова «априорный», которое обозначало распределение $p(\theta)$, описывающее гипотезы о параметрах модели.

Логарифмическое правдоподобие можно представить в виде

$$\log p_{model}(x) = \log \sum_h p_{model}(h,x) \tag{1.3}$$

Мы можем рассматривать автокодировщик как аппроксимацию этой суммы точечной оценкой для всего одного значения h, имеющего высокую вероятность. Это похоже на порождающую модель разреженного кодирования, только h теперь является выходом параметрического кодировщика, а не результатом оптимизации, которая выводит наиболее вероятное значение h. С этой точки зрения, при таком выборе h мы максимизируем функцию

$$\log p_{model}(h,x) = \log p_{model}(h) + \log p_{model}(x|h) \tag{1.4}$$

Член $\log p_{model}(h)$ может индуцировать разреженность. Например, априорное распределение Лапласа

$$p_{model}(h_i) = \frac{\lambda}{2} e^{-\lambda|h_i|} \tag{1.5}$$

соответствует штрафу разреженности по норме L^1 (сумма абсолютных величин).

Действительно,

$$\Omega(h) = \lambda \sum_i |h_i| \tag{1.6}$$

$$-\log p_{model}(h) = \sum_i \left(\lambda|h_i| - \log\frac{\lambda}{2}\right) = \Omega(h) + const \tag{1.7}$$

где постоянный член зависит только от λ и не зависит от h. Обычно λ считается гиперпараметром, а постоянный член отбрасывается, потому что не влияет на обучение параметров. Разреженность могут индуцировать и другие априорные распределения, например t-распределение Стьюдента. При таком взгляде на разреженность как результат влияния $p_{model}(h)$ на обучение с целью аппроксимации максимального правдоподобия штраф разреженности вообще не является регуляризирующим членом. Это просто следствие распределения латентных переменных модели. Такой взгляд дает еще один мотив для обучения автокодировщика: это способ приближенного обучения
порождающей модели. И попутно выясняется еще одна причина полезности признаков, обученных автокодировщиком: они описывают латентные переменные, объясняющие входные данные.

Один из способов достижения нулей в коде h для разреженных (и шумоподавляющих) автокодировщиков использовать блоки линейной ректификации для порождения кодового слоя. Взяв априорное распределение, которое толкает представления в сторону нуля (например, штраф по норме L^1), можно косвенно управлять средним числом нулей в представлении.

1.2.2. Шумоподавляющие автокодировщики

Получить автокодировщик, который обучается чему-то полезному, можно не только путем добавления штрафа в функцию стоимости, но и изменив в ней член реконструкции ошибки.

Традиционно автокодировщики минимизируют некоторую функцию

$$L(x, g(f(x))), \tag{1.8}$$

где L – функция потерь, штрафующая $g(f(x))$ за непохожесть на x, например норма L^2 разности. Это побуждает $g \circ f$ быть просто тождественной функцией, если емкости g и f для этого достаточно.(∘ - функционал – это правило, которое ставит в соответствие элементам одного множества элементы другого множества)

Шумоподавляющий автокодировщик (denoising autoencoder – DAE) вместо этого минимизирует функцию

$$L(x, g(f(\tilde{x}))), \tag{14.9}$$

где $\tilde{x}$ – копия x, искаженная каким-то шумом. Поэтому шумоподавляющий автокодировщик должен скорректировать искажение, а не просто скопировать свой вход.

Обучение с целью шумоподавления принуждает f и g неявно обучиться структуре $p_{data}(x)$. Следовательно, шумоподавляющие автокодировщики – еще один пример того, как полезные свойства могут оказаться побочным результатом минимизации ошибки реконструкции. Вместе с тем это пример того, как повышающую модель с высокой емкостью можно использовать в качестве автокодировщика, если позаботиться о том, чтобы она не обучилась тождественной функции.

1.2.3. Регуляризация посредством штрафования производных

Еще одна стратегия регуляризации автокодировщика подразумевает использование штрафа, как в разреженных автокодировщиках:

$$L(x, g(f(x))) + \Omega(h, x), \tag{1.10}$$

но с Ω другого вида:

$$\Omega(h, x) = \lambda \sum_i \|\nabla_x h_i\|^2 \tag{1.11}$$

Это понуждает модель обучить функцию, которая не сильно изменяется при малом изменении x. Поскольку этот штраф применяется только к обучающим

примерам, он заставляет автокодировщик обучиться признакам, улавливающим информацию об обучающем распределении.

Регуляризированный таким способом автокодировщик называется сжимающим (contractive autoencoder – CAE). Существуют теоретические связи между этим подходом и шумоподавляющими автокодировщиками, обучением многообразий и вероятностным моделированием.

1.3. Репрезентативная способность, размер слоя и глубина

Часто при обучении автокодировщиков используют кодировщик и декодер всего с одним слоем. Но это необязательно. На самом деле глубокие кодировщики и декодеры дают целый ряд преимуществ.

У глубоких сетей прямого распространения есть много достоинств. Поскольку автокодировщики – это сети прямого распространения, то все эти достоинства свойственны и им тоже. Более того, кодировщик и декодер по отдельности тоже являются сетями прямого распространения, поэтому каждая компонента автокодировщика может получить выигрыш от глубины.

Важное преимущество нетривиальной глубины состоит в том, что согласно универсальной теореме аппроксимации нейронной сетью прямого распространения хотя бы с одним скрытым слоем гарантированно можно аппроксимировать любую функцию (из весьма широкого класса) с произвольной точностью, при условии, что количество скрытых блоков достаточно велико. Это означает, что автокодировщик с одним скрытым слоем способен представить тождественную функцию в области определения данных с произвольной точностью. Однако отображение входа на код мелкое, т. е. мы не можем наложить произвольных ограничений, например потребовать, чтобы код был разреженным. Глубокий автокодировщик, имеющий, по крайней мере, один дополнительный скрытый слой внутри самого кодировщика, может аппроксимировать любое отображение входа на код с произвольной точностью при наличии достаточного числа скрытых блоков

Увеличение глубины может экспоненциально уменьшить вычислительную стоимость представления некоторых функций, а также объем данных, необходимых для обучения некоторых функций. Экспериментально показано, что глубокие автокодировщики достигают гораздо более высокой степени сжатия, чем соответствующие мелкие или линейные автокодировщики.

Общая стратегия обучения глубокого автокодировщика состоит в жадном предобучении глубокой архитектуры посредством обучения ряда мелких автокодировщиков, именно поэтому часто встречаются мелкие автокодировщики даже тогда, когда целью является обучение глубокого.

1.4. Стохастические кодировщики и декодеры

Автокодировщики – это просто сети прямого распространения. Те же функции потерь и типы выходных блоков, что применяются в традиционных сетях прямого распространения, можно использовать и в автокодировщиках.

Тогда общая стратегия проектирования выходных блоков и функции потерь в сети прямого распространения заключается в том, чтобы определить выходное распределение $p(y|x)$ и минимизировать отрицательное логарифмическое правдоподобие $-\log p(y|x)$. В такой постановке y – это вектор целей, например меток классов.

В автокодировщике x является не только входом, но и целью. Можно считать, что декодер дает условное распределение $p_{decoder}(x|h)$ при условии скрытого кода h. Тогда можно обучить автокодировщик путем минимизации $-\log p_{decoder}(x|h)$. Точная форма этой функции потерь будет зависеть от формы декодера. Как и в традиционных сетях прямого распространения, когда x принимает вещественные значения, мы обычно используем линейные выходные блоки для параметризации среднего нормального распределения. В этом случае отрицательное логарифмическое правдоподобие дает критерий среднеквадратической ошибки. Аналогично бинарным значениям x соответствует распределение Бернулли, параметры которого задаются сигмоидным выходным блоком, дискретным значениям x – softmax-распределение и т. д. Как правило, выходные переменные считаются условно независимыми при условии h, чтобы это распределение вероятности было проще вычислить, но существуют методы, например выход в виде смеси распределений, которые позволяют моделировать коррелированный выход с приемлемыми вычислительными затратами.

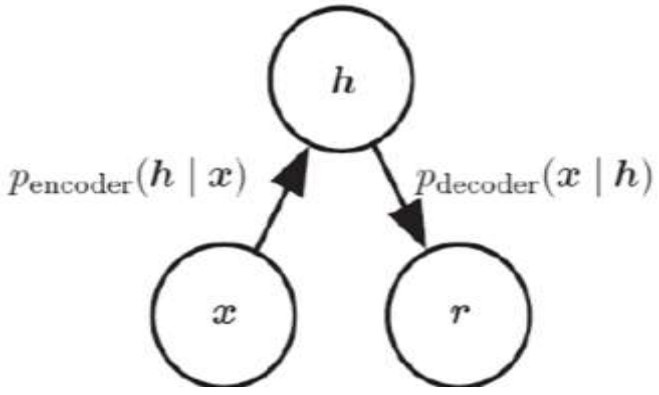

Рис. 1.2 Структура стохастического автокодировщика, в которой кодировщик и декодер – функции, содержащие шум, т. е. их выход можно рассматривать как выборку из распределения $p_{encoder}(h|x)$ для кодировщика и $p_{decoder}(x|h)$ для декодера

Любая модель с латентными переменными $p_{model}(h, x)$ определяет стохастический кодировщик

$$p_{encoder}(h|x) = p_{model}(h|x) \quad (1.12)$$

и стохастический декодер

$$p_{decoder}(x|h) = p_{model}(x|h). \quad (1.13)$$

В общем случае распределения кодировщика и декодера не обязательно являются условными распределениями, совместимыми с совместным распределением $p_{model}(x, h)$. Определено, что обучение кодировщика и декодера как шумоподавляющего автокодировщика делает их асимптотически совместимыми (при достаточной емкости и количестве обучающих примеров).

1.5. Шумоподавляющие автокодировщики

Шумоподавляющим автокодировщиком (DAE) называется автокодировщик, который получает на входе искаженные данные и обучается предсказывать истинные, неискаженные данные.

Процедура обучения DAE показана на рис. 1.3.

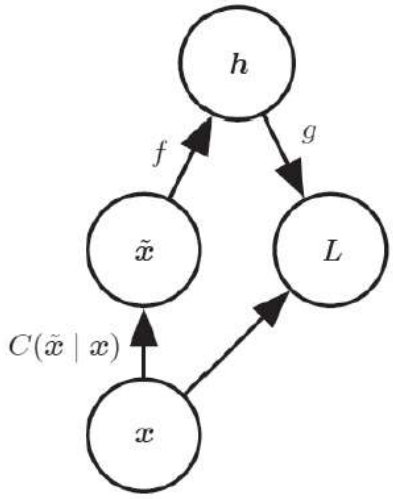

Рис. 1.3 Граф вычислений функции стоимости для шумоподавляющего автокодировщика, обученного реконструировать чистые данные x по искаженным $\tilde{x}$. Это достигается путем минимизации потери $L = -\log p_{decoder}(x|h = f(\tilde{x}))$, где $\tilde{x}$ – искаженная версия примера x, полученная посредством заданного искажающего процесса $C(\tilde{x}|x)$. Обычно $p_{decoder}$ является факторным распределением, средние параметры которого порождаются сетью прямого распространения g

Мы вводим искажающий процесс $C(\tilde{x}|x)$, который представляет условное распределение искаженных примеров $\tilde{x}$ при условии истинных примеров x. Затем

автокодировщик обучается распределению реконструкции $p_{reconstruct}(x|\tilde{x})$, которое оценивается по обучающим парам $(\tilde{x}|x)$ следующим образом:

1) выбрать обучающий пример x из обучающих данных;

2) выбрать искаженную версию $\tilde{x}$ из $C(\tilde{\mathrm{x}}|\mathrm{x} = x)$;

3) использовать $(\tilde{x}|x)$ в качестве обучающего примера для оценки распределения реконструкции автокодировщика $p_{reconstruct}(x|\tilde{x}) = p_{decoder}(x|h)$, где h – выход кодировщика $f(\tilde{x})$, а $p_{decoder}$ обычно определяется декодером $g(h)$.

Как правило, мы можем просто выполнить приближенную градиентную минимизацию (например, мини-пакетный градиентный спуск) отрицательного логарифмического правдоподобия $L = -\log p_{decoder}(x|h)$. Если кодировщик детерминирован, то шумоподавляющий автокодировщик является сетью прямого распространения, и для его обучения можно применить все те же методы, что для любой сети прямого распространения.

Таким образом, мы можем считать, что DAE применяет метод стохастического градиентного спуска к следующему математическому ожиданию:

$$-\mathbb{E}_{x\sim\hat{p}_{data}(x)}\mathbb{E}_{\tilde{x}\sim C(\tilde{x}|x)}\log p_{decoder}(x|h = f(\tilde{x}))\,, \qquad (1.14)$$

где $\hat{p}_{data}(x)$ – распределение обучающих данных.

1.5.1. Сопоставление рейтингов

Сопоставление рейтингов (score matching) – альтернатива максимальному правдоподобию. Этот метод дает состоятельную оценку распределений вероятности, поощряя модель иметь такой же рейтинг (score), как распределение данных на каждом обучающем примере x. В этом контексте рейтингом является конкретное поле градиента:

$$\nabla_x \log p(x) \qquad (1.15)$$

Здесь достаточно понимать, что обучения поля градиента $\log p_{data}$ – один из способов обучиться структуре самого распределения p_{data}.

У шумоподавляющих автокодировщиков есть очень важное свойство: критерий их обучения (с условно нормальным $p(x|h)$) таков, что автокодировщик обучается векторному полю $(g(f(x))-x)$, являющемуся оценкой для рейтинга распределения данных. Это показано на рис. 1.4.

Шумоподавляющее обучение автокодировщиков конкретного вида (с сигмоидными скрытыми блоками и линейными блоками реконструкции) с использованием гауссова шума и среднеквадратической ошибки в качестве стоимости реконструкции эквивалентно обучению специального вида неориентированной вероятностной модели – ограниченной машины Больцмана (ОМБ) с гауссовыми видимыми блоками. Мы будем рассматривать эту модель подробно сейчас нам достаточно знать, что она дает явное распределение

$p_{model}(x;\theta)$. Если ОМБ обучается с применением шумоподавляющего сопоставления рейтингов, то алгоритм обучения эквивалентен шумоподавляющему обучению в соответствующем автокодировщике. При фиксированном уровне шума регуляризированное сопоставление рейтингов не является состоятельной оценкой, оно восстанавливает размытую версию распределения. Но если уровень шума стремится к 0, когда число примеров стремится к бесконечности, то состоятельность восстанавливается.

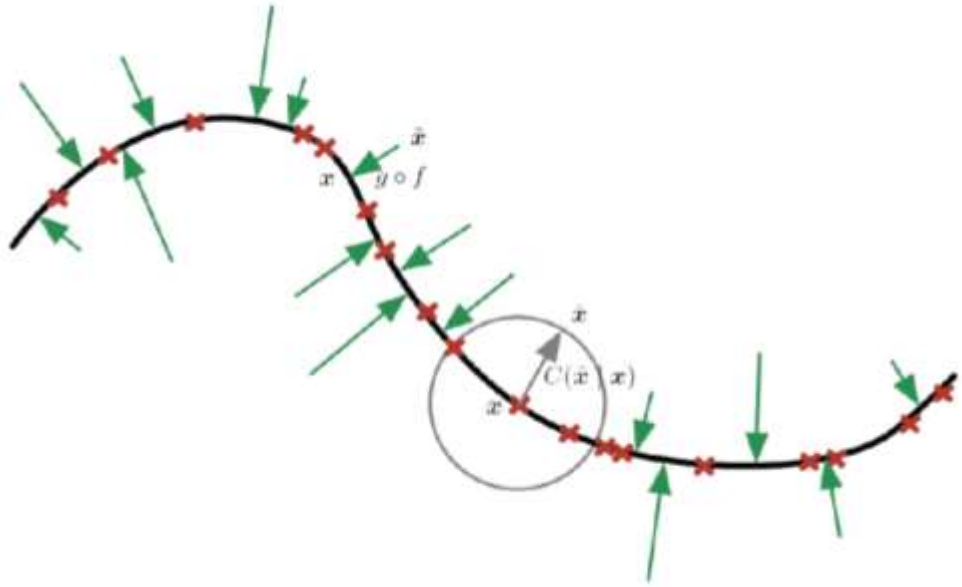

Рис. 1.4 Шумоподавляющий автокодировщик обучается отображать искаженные данные $\tilde{x}$ на исходные x. Обучающие примеры x обозначены красными крестиками, лежащими в окрестности многообразия низкой размерности, показанного жирной черной линией. Искажающий процесс $C(\tilde{x}|x)$ представлен серой окружностью с равновероятными искажениями. Серая стрелка показывает, как один обучающий пример преобразуется в результат одной выборки из искажающего процесса. Когда шумоподавляющий автокодировщик обучается минимизировать среднее квадратичных ошибок $||g(f(\tilde{x}))-x||^2$, реконструкция $g(f(\tilde{x}))$ оценивает $\mathbb{E}_{x,\tilde{x}\sim p_{data}(x)C(\tilde{x}\mid x)}[x \mid \tilde{x}]$. Вектор $g(f(\tilde{x}\sim))-\tilde{x}$ направлен приблизительно в сторону ближайшей точки на многообразии, поскольку $g(f(\tilde{x}))$ оценивает центр тяжести неискаженных точек x, которые могли бы привести к $\tilde{x}$. Таким образом, автокодировщик обучается векторному полю $g(f(x))-x$, обозначенному зелеными стрелками. Это векторное поле является оценкой рейтинга $\nabla_x \log p_{data}(x)$ с точностью до множителя, равного среднеквадратической ошибке реконструкции.

Существуют и другие связи между автокодировщиками и ОМБ (ограниченной машины Больцмана). Сопоставление рейтингов в применении к

ОМБ дает функцию стоимости, идентичную ошибке реконструкции в сочетании с регуляризирующим членом, аналогичным сжимающему штрафу CAE (сжимающий автокодировщик). Исследовано, что градиент автокодировщика дает аппроксимацию обучения ОМБ методом сопоставительного расхождения.

Для непрерывных x шумоподавляющий критерий с гауссовым искажением и распределением реконструкции дает оценку рейтинга, применимую к общим параметризациям кодировщика и декодера. Это означает, что общей архитектурой кодировщик-декодер можно воспользоваться для оценивания рейтинга, если проводить обучение с критерием квадратичной ошибки

$$||g(f(\tilde{x}))-x||^2 \tag{1.16}$$

и искажающим процессом

$$C(\tilde{x}=\tilde{x}|x)=\mathcal{N}(\tilde{x};\ \mu=x,\Sigma=\sigma^2 I) \tag{1.17}$$

дисперсией шума σ^2. Принцип работы иллюстрируется на рис. 1.5.

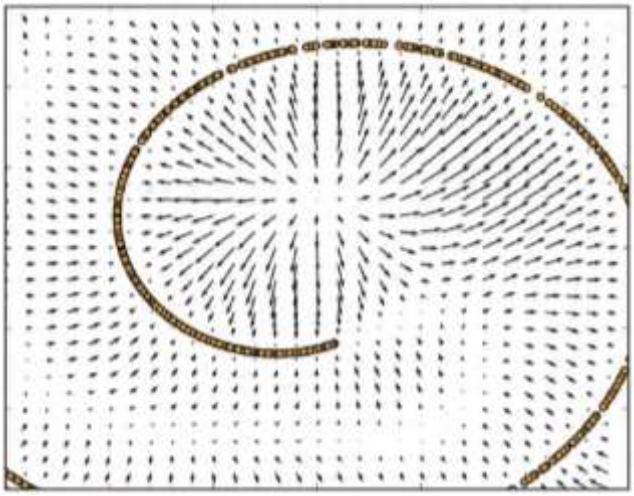

Рис. 1.5 Обученное шумоподавляющим автокодировщиком векторное поле вокруг одномерного искривленного многообразия, в окрестности которого сконцентрированы двумерные данные. Каждая стрелка по длине пропорциональна реконструкции минус входной вектор автокодировщика и направлена в сторону увеличения вероятности в соответствии с неявной оценкой распределения вероятности. Векторное поле обращается в нуль в точках максимума и минимума оцененной функции плотности (на многообразии). Так, отрезок спирали образует одномерное многообразие соединенных друг с другом локальных максимумов. Локальные минимумы находятся вблизи середины разрыва между двумя отрезками. Если норма ошибки реконструкции (пропорциональная длинам стрелок) велика, то вероятность можно значительно повысить, двигаясь в направлении стрелки; такое бывает в основном на участках низкой вероятности. Автокодировщик отображает такие точки с низкой вероятностью на реконструкции, имеющие более высокую вероятность. Там, где

вероятность максимальна, стрелка укорачивается, поскольку реконструкция становится более точной.

В общем случае не гарантируется, что реконструкция $g(f(x))$ минус вход x соответствует градиенту хоть какой-нибудь функции, не говоря уже о рейтинге. Именно поэтому ранние результаты специализированы для конкретных параметризаций, когда $g(f(x)) - x$ можно получить в виде производной какой-то другой функции. Результат может быть обобщен путем идентификации такого семейства мелких автокодировщиков, что $g(f(x)) - x$ соответствует рейтингу для всех членов этого семейства.

До сих пор мы описывали только, как шумоподавляющий автокодировщик обучается представлять распределение вероятности. В более общем случае можно использовать автокодировщик как порождающую модель и делать выборку из этого распределения.

1.6. Обучение многообразий с помощью автокодировщиков

Как и во многих других алгоритмах машинного обучения, в автокодировщиках используется гипотеза о концентрации данных в окрестности многообразия низкой размерности или небольшого множества таких многообразий.

Многообразие – это связная область. С точки зрения математики, это множество точек, ассоциированных с окрестностью каждой точки. Из любой точки многообразие локально выглядит как евклидово пространство. В повседневной жизни мы воспринимаем поверхность земли как двумерную плоскость, но на самом деле это сферическое многообразие в трехмерном пространстве. Из идеи окрестности каждой точки вытекает существование преобразований для перемещения из одного места многообразия в другое. В примере земной поверхности как многообразия мы можем пойти на север, юг, восток и запад.

В некоторых алгоритмах эта идея ограничена лишь обучением функций, которые корректно ведут себя на многообразии, но поведение может оказаться необычным, если предъявить пример, лежащий вне многообразия. Автокодировщики идут дальше и стремятся обучиться структуре многообразия.

Чтобы понять, как они это делают, необходимо познакомиться с некоторыми характеристиками многообразий.

Важной характеристикой является множество касательных плоскостей. Касательная плоскость в точке x на d-мерном многообразии определяется d базисными векторами в локальных направлениях допустимых изменений. Как

показано на рис. 1.6, эти локальные направления описывают, какие возможны бесконечно малые изменения x, не выводящие за пределы многообразия.

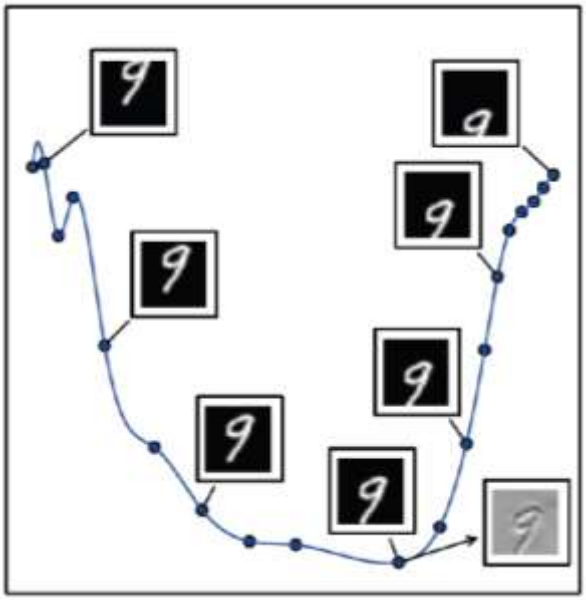

Рис. 1.6 Иллюстрация понятия касательной гиперплоскости. Здесь создается одномерное многообразие в 784-мерном пространстве. Мы берем изображение из набора данных MNIST, содержащее 784 пикселя, и преобразуем его, выполняя параллельный перенос по вертикали. Величина переноса определяет координату вдоль одномерного многообразия которая описывает искривленный путь в пространстве изображения. На графике показано несколько точек на этом многообразии. Для наглядности мы спроецировали многообразие на двумерное пространство методом PCA (метод главных компонент). У n-мерного многообразия в каждой точке имеется n-мерная касательная плоскость. Эта плоскость имеет с многообразием ровно одну общую точку и ориентирована параллельно его поверхности в этой точке. Она определяет направления, в которых можно двигаться, оставаясь на многообразии. У данного одномерного многообразия касательная плоскость вырождается в прямую. Мы привели пример касательной прямой в одной точке вместе с изображением, показывающим, как ее направление выглядит в пространстве изображения. Серым цветом обозначены пиксели, которые не изменяются при движении вдоль касательной, белым – пиксели, которые становятся ярче, а черным – те, что становятся темнее

Любая процедура обучения автокодировщика включает компромисс между двумя стремлениями.

1. Обучить такое представление h обучающего примера x, чтобы x можно было приближенно восстановить по h с помощью декодера. Тот факт, что x выбирается из обучающих данных, критически важен, поскольку означает, что

автокодировщик не обязан успешно реконструировать входы, не вероятные с точки зрения порождающего данные распределения.

2. Удовлетворить ограничение или регуляризирующий штраф. Это может быть архитектурное ограничение, ограничивающее емкость автокодировщика, или регуляризирующий член, прибавленный к стоимости реконструкции. Смысл в любом случае – отдавать предпочтение решениям, менее чувствительным к входу.

Очевидно, что поодиночке оба стремления бесполезны: ни копирование входа в выход, ни полное игнорирование входа нам ни к чему. Но вместе они становятся осмысленными, потому что вынуждают скрытое представление уловить информацию о структуре порождающего данные распределения. Важный принцип заключается в том, что автокодировщик может ограничиться представлением только тех изменений, которые необходимы для реконструкции обучающих примеров. Если порождающее распределение концентрируется вблизи многообразия низкой размерности, то получатся представления, которые неявно улавливают локальную систему координат этого многообразия: лишь изменения, касательные к многообразию в точке x, должны соответствовать изменениям $h = f(x)$. Следовательно, кодировщик обучается такому отображению из пространства входов x в пространство представления, которое чувствительно только к изменениям вдоль направлений, касательных к многообразию, и нечувствительно к изменениям вдоль ортогональных к многообразию направлений.

На рис. 1.7 приведен одномерный пример, показывающий, что, сделав функцию реконструкции нечувствительной к возмущениям входа в окрестности данных, мы заставили автокодировщик восстановить структуру многообразия.

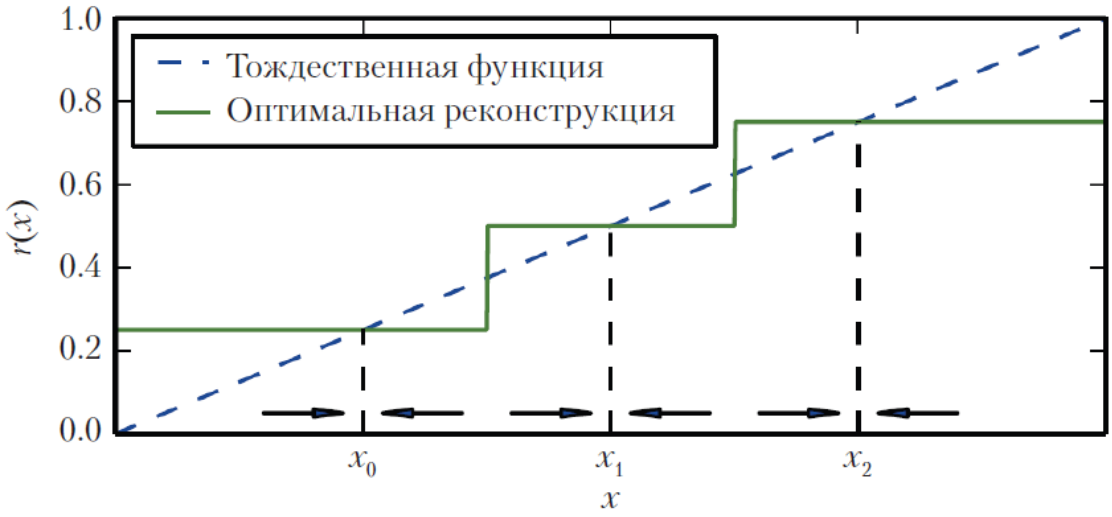

Рис. 1.7 Если автокодировщик обучает функцию реконструкции, инвариантную к малым возмущениям в окрестности входных точек, то он улавливает структуру

многообразия, на котором лежат данные. В данном случае структурой является набор 0-мерных многообразий. Штриховая диагональная прямая обозначает тождественную целевую функцию, подлежащую реконструкции. Оптимальная функция реконструкции пересекает график тождественной функции в каждой точке данных. Горизонтальные стрелки в нижней части рисунка обозначают вектор направления реконструкции $r(x)-x$ в пространстве входов и всегда указывают в сторону ближайшего «многообразия» (единственной точки в одномерном случае). Шумоподавляющий автокодировщик пытается сделать производную функции реконструкции $r(x)$ малой в окрестности входных точек. Сжимающий автокодировщик делает то же самое для кодировщика. Хотя производную $r(x)$ понуждают быть малой вблизи входных точек, между ними она может быть большой. Пространство между входными точками соответствует области между многообразиями, где функция реконструкции должна иметь большую производную, чтобы вернуть искаженные точки на многообразие

Чтобы понять, почему автокодировщики полезны для обучения многообразий, поучительно сравнить их с другими подходами. Для характеристики многообразия обычно обучают представление точек данных на многообразии (или вблизи него). Для конкретного примера такое представление называют также его погружением. Как правило, оно описывается вектором более низкой размерности, чем у объемлющего пространства, подмножеством которого является многообразие. Некоторые алгоритмы непосредственно обучают погружение для каждого обучающего примера, тогда как другие обучают более общее отображение, иногда называемое кодировщиком, или функцией представления, которое переводит любую точку объемлющего пространства (пространства входов) в ее погружение.

Обучение многообразий по большей части включает в себя процедуры обучения без учителя, пытающиеся эти многообразия уловить. В большинстве ранних работ по обучению нелинейных многообразий акцент ставился на непараметрические методы, основанные на графе ближайших соседей. В этом графе имеется одна вершина для каждого обучающего примера и ребра, соединяющие ближайших соседей. Эти методы связывают с каждой вершиной касательную плоскость, натянутую на направления изменения, определяемые векторами разностей между примером и его соседями, как показано на рис. 14.8. Глобальную систему координат можно затем получить с помощью оптимизации или путем решения системы линейных уравнений.

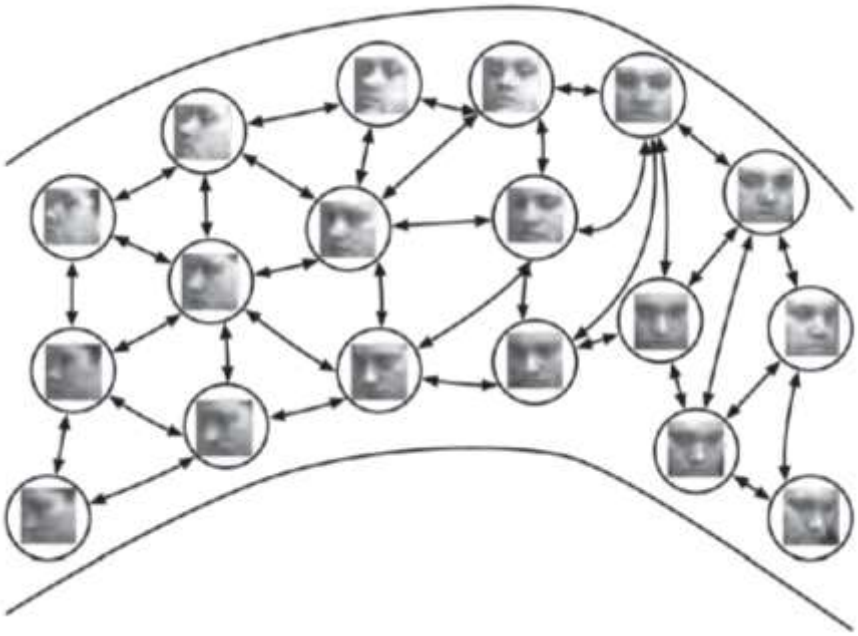

Рис. 1.8 Непараметрические процедуры обучения многообразий строят граф ближайших соседей, вершины которого представляют обучающие примеры, а ориентированные ребра – связи с ближайшими соседями. Существуют различные процедуры, позволяющие получить касательную плоскость, ассоциированную с соседством в этом графе, а также систему координат, которая ассоциирует каждый обучающий пример с положением вещественного вектора, или погружением. Такое представление обобщается на новые примеры с помощью интерполяции. Если число примеров достаточно велико для покрытия кривизны и скручивания многообразия, то такие методы работают хорошо.

На рис. 1.9 показано, как можно замостить многообразие большим числом локально линейных «блинов», напоминающих гауссианы, плоские в касательных направлениях.

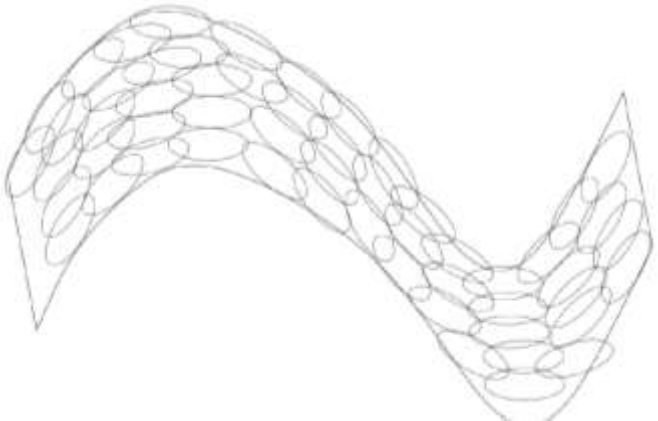

Рис. 1.9 Если касательные плоскости (см. рис. 1.6) в каждой точке известны, то из них можно образовать глобальную систему координат, или функцию плотности. Каждую локальную «плитку» можно рассматривать как локальную

евклидову систему координат или как локально плоское нормальное распределение («блин») с очень малой дисперсией в направлениях, ортогональных блину, и очень большой в направлениях, определяющих систему координат блина. Смесь таких нормальных распределений дает оценку функции плотности.

Фундаментальная трудность таких локальных непараметрических подходов к обучению многообразий описана следующим образом: если многообразие не слишком гладкое (имеет много пиков, впадин и скручиваний), то для покрытия всех вариаций может понадобиться очень много обучающих примеров, но шансов обобщения на ранее не предъявлявшиеся вариации все равно не будет. Действительно, эти методы способны обобщить форму многообразия только путем интерполяции соседних примеров. К сожалению, многообразия, встречающиеся в задачах ИИ, могут иметь очень сложную структуру, которую трудно уловить одной лишь локальной интерполяцией. Рассмотрим пример многообразия, полученного в результате параллельных переносов, на рис. 1.6. Если мы наблюдаем только одну координату входного вектора x_i по мере переноса изображения, то будем видеть максимум или минимум этой координаты всякий раз, как в яркости изображения встречается пик или впадина. Иными словами, сложность паттернов яркости в исходном изображении определяет сложность многообразий, порождаемых его простыми преобразованиями. Это наводит на мысль об использовании распределенных представлений и глубокого обучения для улавливания структуры многообразия.

1.7. Сжимающие автокодировщики

Сжимающий автокодировщик (CAE) вводит явный регуляризирующий член для кода $h = f(x)$, поощряющий за небольшую величину производных f:

$$\Omega(h,) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2 \qquad (1.18)$$

Штраф $\Omega(h)$ равен квадрату нормы Фробениуса (сумма квадратов элементов) матрицы Якоби, состоящей из частных производных функции кодирования.

Существует связь между шумоподавляющим и сжимающим автокодировщиками: определено, что в пределе малого гауссова входного шума ошибка реконструкции шумоподавляющего автокодировщика эквивалентна сжимающему штрафу в функции реконструкции, которая отображает x в $r = g(f(x))$. Иными словами, шумоподавляющий автокодировщик понуждает функцию реконструкции сопротивляться малым, но конечным возмущениям

входа, тогда как сжимающий автокодировщик заставляет функцию выделения признаков сопротивляться бесконечно малым возмущениям входа. Когда сжимающий штраф на основе матрицы Якоби применяется для предобучения признаков $f(x)$ с целью последующего использования в классификаторе, наилучшая верность классификации обычно получается, если применять штраф к $f(x)$, а не к $g(f(x))$. Сжимающий штраф в $f(x)$ также имеет тесные связи с сопоставлением рейтингов.

Термин сжимающий объясняется тем, как CAE (сжимающий автокодировщик) деформирует пространство. Поскольку CAE обучен противиться возмущениям входа, то поощряется отображение окрестности входной точки в меньшую окрестность выходной точки. Это можно рассматривать как сжатие окрестности входной точки.

Уточним, что CAE является сжимающим только локально – все возмущения точки x обучающего набора отображаются в малую окрестность $f(x)$. Глобально образы $f(x)$ и $f(x')$ точек x и x' могут отстоять друг от друга дальше, чем исходные точки. Вполне может случиться, что между многообразиями или вдали от них функция f будет не сжимающей, а, наоборот, расширяющей (см., например, что происходит в тривиальном одномерном примере на рис. 1.7). Если штраф $\Omega(h)$ применяется к сигмоидным блокам, то для сжатия якобиана достаточно потребовать, чтобы эти блоки асимптотически приближались к 0 или 1. Тогда штраф поощряет CAE (сжимающий автокодировщик) кодировать входные точки экстремальными значениями сигмоиды, что можно интерпретировать как двоичный код. Также гарантируется, что CAE будет рассредоточивать кодовые значения по большей части гиперкуба, покрываемого его сигмоидными скрытыми блоками.

Мы можем рассматривать матрицу Якоби J в точке x как аппроксимацию нелинейного кодировщика $f(x)$ линейным оператором. Это позволяет формализовать слово «сжимающий». В теории линейных операторов говорят, что линейный оператор сжимающий, если норма Jx меньше или равна 1 для всех векторов x с единичной нормой. Иначе говоря, J сжимающий, если он стягивает единичную сферу. Можно считать, что CAE штрафует норму Фробениуса локальной линейной аппроксимации $f(x)$ в каждой точке x обучающего набора, стремясь сделать все такие локальные линейные операторы сжимающими

Как описано в разделе 1.6, регуляризированные автокодировщики обучают многообразия, балансируя под действием двух противоположно направленных сил. В случае CAE в роли таких сил выступают ошибка реконструкции и сжимающий штраф $\Omega(h)$. Одна лишь ошибка реконструкции поощряла бы CAE

обучиться тождественной функции, а один лишь сжимающий штраф – обучиться признакам, постоянным относительно x. В результате компромисса между тем и другим получается автокодировщик, у которого производные $\partial f(x)/\partial x$ в основном очень малы. И лишь для немногих скрытых блоков, которые соответствуют небольшому числу направлений во входных данных, производные могут быть велики.

Цель CAE – обучиться структуре многообразия данных. В направлениях x с большими значениями Jx вектор h быстро изменяется, поэтому они, вероятно, являются направлениями, аппроксимирующими касательные плоскости к многообразию. Эксперименты показывают, что обучение CAE (сжимающего автокодировщика) приводит к тому, что в большинстве своем сингулярные числа J по абсолютной величине меньше 1, т. е. являются сжимающими. Но некоторые сингулярные числа все же оказываются больше 1, поскольку штраф за ошибку реконструкции поощряет CAE кодировать направления с наибольшей локальной дисперсией. Направления, соответствующие наибольшим сингулярным числам, интерпретируются как касательные направления, обученные сжимающим автокодировщиком. В идеале они должны соответствовать реальной вариативности данных. Например, в случае применения к изображениям CAE должен обучиться касательным векторам, которые показывают, как изменяется изображение, когда присутствующие в нем объекты постепенно меняют расположение, как на рис. 1.6. Визуализация экспериментально полученных сингулярных векторов, похоже, действительно соответствует осмысленным преобразованиям входного изображения, как видно по рис. 1.10.

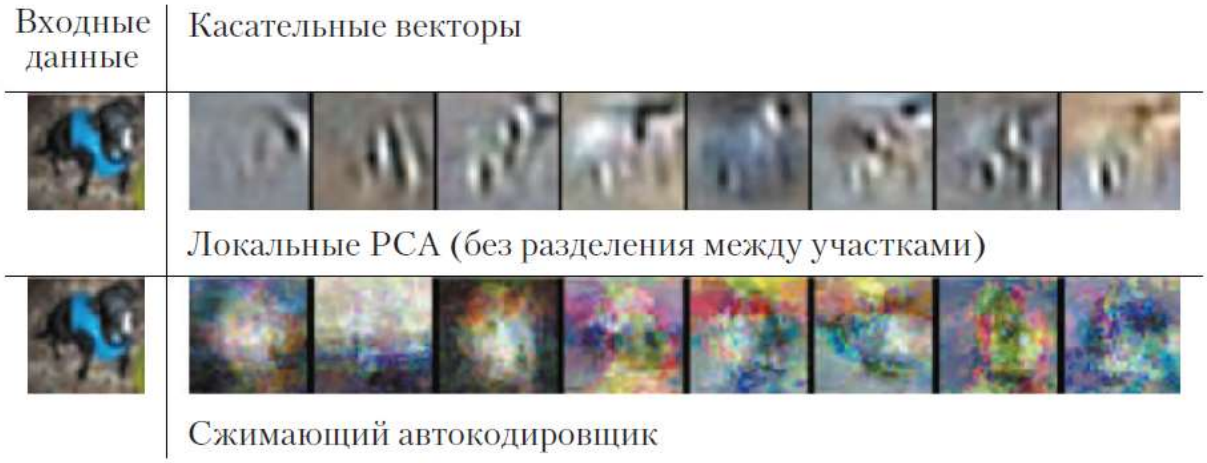

Рис. 14.10 Касательные векторы к многообразию, оцененные локальным методом главных компонент (PCA) и сжимающим автокодировщиком. Позиция на многообразии определяется входным изображением собаки, взятым из набора данных CIFAR-10. Для оценки касательных векторов используются первые

сингулярные векторы матрицы Якоби $\partial h/\partial x$ отображения входа на код. Хотя и PCA (методо главных компонент), и CAE (сжимающего автокодировщика) могут найти локальные касательные, CAE дает более точные оценки на ограниченном объеме обучающих данных, поскольку в нем присутствует разделение параметров между разными позициями, совместно использующими некоторое подмножество активных скрытых блоков. Касательные направления, найденные CAE, обычно соответствуют перемещающимся или изменяющимся частям объекта (скажем, голове или лапам).

С критерием регуляризации CAE возникает одна практическая проблема: его вычисление в случае автокодировщика с одним скрытым слоем обходится дешево, но становится гораздо дороже, когда слоев больше. Можно применить следующая стратегия – последовательность однослойных автокодировщиков обучается так, чтобы каждый следующий обучался реконструировать скрытый слой предыдущего. Их композиция и образует глубокий автокодировщик. Поскольку каждый слой локально сжимающий, то и глубокий автокодировщик тоже будет сжимающим. Результат получается не таким же, как при совместном обучении всей архитектуры со штрафом на якобиан глубокой модели, но многие желательные качественные характеристики улавливаются.

Еще одна практическая проблема состоит в том, что применение сжимающего штрафа может давать бесполезные результаты, если не задать какого-то масштаба декодера. Например, действие кодировщика может заключаться в умножении входа на небольшую константу ε, а действие декодера – в делении кода на ε. Когда ε стремится к 0, кодировщик устремляет сжимающий штраф $\Omega(h)$ к 0, так ничего и не узнав о распределении. А тем временем декодер демонстрирует идеальную реконструкцию. Для предотвращения такой ситуации веса f и g связываются. И f, и g – стандартные слои нейронной сети, состоящие из аффинного преобразования с последующей поэлементной нелинейностью, поэтому можно просто сделать матрицу весов g транспонированной к матрице весов f.

1.8. Предсказательная разреженная декомпозиция

Предсказательная разреженная декомпозиция (ПРД) (predictive sparse decomposition – PSD) – гибридная модель, в которой сочетаются разреженное кодирование и параметрические автокодировщики. Параметрический кодировщик обучается предсказывать результат итеративного вывода. ПРД применялась к обучению признаков без учителя для распознавания объектов в изображениях и видео, а также к обработке звуковой информации. Модель включает кодировщик

$f(x)$ и декодер $g(h)$ – оба параметрические. В процессе обучения h контролируется алгоритмом оптимизации. Обучение заключается в минимизации

$$||x - g(h)||^2 + \lambda|h|_1 + \gamma||h - f(x)||^2 \quad (1.19)$$

Как в случае разреженного кодирования, алгоритм обучения чередует минимизацию относительно h с минимизацией относительно параметров модели. Минимизация относительно h производится быстро, потому что $f(x)$ дает хорошее начальное значение h, а функция стоимости налагает на h ограничение – оставаться близко к $f(x)$. Простым градиентным спуском можно получить разумные значения h всего за десять шагов.

Процедура обучения в ПРД (предсказательной разреженной декомпозиции) не сводится к обучению сначала модели разреженного кодирования, а затем обучению $f(x)$ для предсказания значений признаков разреженного кодирования. На самом деле эта процедура регуляризирует декодер, так чтобы использовались параметры, для которых $f(x)$ может вывести хорошие значения кода.

Предсказательная разреженная декомпозиция – пример обученного приближенного вывода. Из этого следует, что ПРД можно интерпретировать как обучение ориентированной вероятностной модели разреженного кодирования путем максимизации нижней границы логарифмического правдоподобия модели.

В практических приложениях ПРД итеративная оптимизация используется только на этапе обучения. Параметрический кодировщик f применяется для вычисления обученных признаков после развертывания модели. Вычисление f обходится недорого, по сравнению с выводом h методом градиентного спуска. Поскольку f – дифференцируемая параметрическая функция, то модели ПРД можно объединить и использовать для инициализации глубокой сети, обучаемой по другому критерию.

1.9. Применения автокодировщиков

Автокодировщики успешно применялись в задачах понижения размерности и информационного поиска. Понижение размерности было одним из первых приложений обучения представлений и глубокого обучения, а также одним из ранних стимулов к изучению автокодировщиков. Например, был обучен стек ОМБ (ограниченная машина Больцмана), а затем их веса использовались для инициализации глубокого автокодировщика с постепенно уменьшающимися скрытыми слоями, так что в последнем слое было всего 30 блоков. Получившийся код давал меньшую ошибку реконструкции, чем PCA (метод главных компонент), в 30 направлениях, а обученное представление оказалось проще качественно интерпретировать и связать с объясняющими категориями, которые проявлялись в виде четко разделенных кластеров.

Представления низкой размерности могут повысить качество во многих задачах, в т.ч. классификации. Чем меньше пространство, тем меньше памяти занимает модель и тем быстрее производятся расчеты. Многие способы понижения размерности помещают семантически похожие примеры рядом. Информация, сохраняемая при отображении в пространство меньшей размерности, способствует лучшей обобщаемости.

Особенно сильно выигрывает от понижения размерности информационный поиск – задача об отыскании в базе данных записей, похожих на указанную в запросе. Дополнительный выигрыш связан с тем, что в некоторых видах пространств низкой размерности поиск может оказаться чрезвычайно эффективен. Точнее, если обучить алгоритм понижения размерности порождать двоичный код низкой размерности, то мы сможем поместить всю базу в хэш-таблицу, которая отображает двоичные кодовые векторы на полные записи. Тогда результатом информационного поиска будет множество всех записей базы данных с таким же двоичным кодом, как у запрошенной. Мы можем также очень эффективно искать чуть менее похожие данные, для чего нужно просто инвертировать отдельные биты в кодовом представлении запроса. Такой подход к информационному поиску посредством понижения размерности и бинаризации называется семантическим хэшированием, он успешно применялся к поиску как текстов, так и изображений.

Для порождения двоичных кодов семантического хэширования обычно используется функция кодирования с сигмоидными блоками в последнем слое. Эти блоки необходимо обучить так, чтобы они были асимптотически близки к 0 или к 1 для всех входных значений. Один из способов добиться этого – внести аддитивный шум перед сигмоидной нелинейностью во время обучения. Абсолютная величина шума должна возрастать со временем. Чтобы противостоять шуму и сохранить как можно больше информации, сеть должна увеличивать величину входов сигмоидной функции до наступления насыщения.

Идея обучения хэш-функции исследовалась в нескольких направлениях, в т.ч. для обучения оптимизирующих потерю представлений, которые в большей мере ориентированы на поиск близких примеров в хэш-таблице.