Содержание

РАЗДЕЛ 1
МАТЕМАТИЧЕСКИЕ ОСНОВЫ ГЛУБОКОГО ОБУЧЕНИЯ

Лекция 1.1 - Сбор данных для глубокого обучения

С наступлением эры «**больших данных**» машинное обучение значительно упростилось, поскольку ключевая проблема статистического оценивания – высокое качество обобщения на новые данные после обучения на небольшом количестве примеров – теперь далеко не так актуальна.

Многие алгоритмы машинного обучения резко усложняются, когда размерность данных велика. Это явление называется **проклятием размерности**.

С увеличением размерности данных (слева направо) количество представляющих интерес конфигураций растет экспоненциально. В одномерном случае имеется одна переменная, и для нее всего 10 интересных областей. При достаточном числе примеров, попадающих в каждую область (на рисунке область соответствует одной клетке), от алгоритма обучения легко добиться правильного обобщения. Самый прямолинейный способ – оценить значение метки в каждой области с возможной интерполяцией между соседними областями). В общем случае если имеется d измерений и нужно различать v значений вдоль каждой оси, то потребуется $O(v^d)$ областей и примеров. Это и есть проклятие размерности.

Лекция 1.1 - Сбор данных для глубокого обучения

Можно ли эффективно представить сложную функцию и может ли оцененная функция хорошо обобщаться на новые данные?

Основная идея глубокого обучения состоит в том, что данные предположительно были порождены композицией факторов, или признаков, возможно, организованных иерархически.

ПОПОЛНЕНИЕ НАБОРА ДАННЫХ

Можно ли эффективно представить сложную функцию и может ли оцененная функция хорошо обобщаться на новые данные?

Основная идея глубокого обучения состоит в том, что данные предположительно были порождены композицией факторов, или признаков, возможно, организованных иерархически.

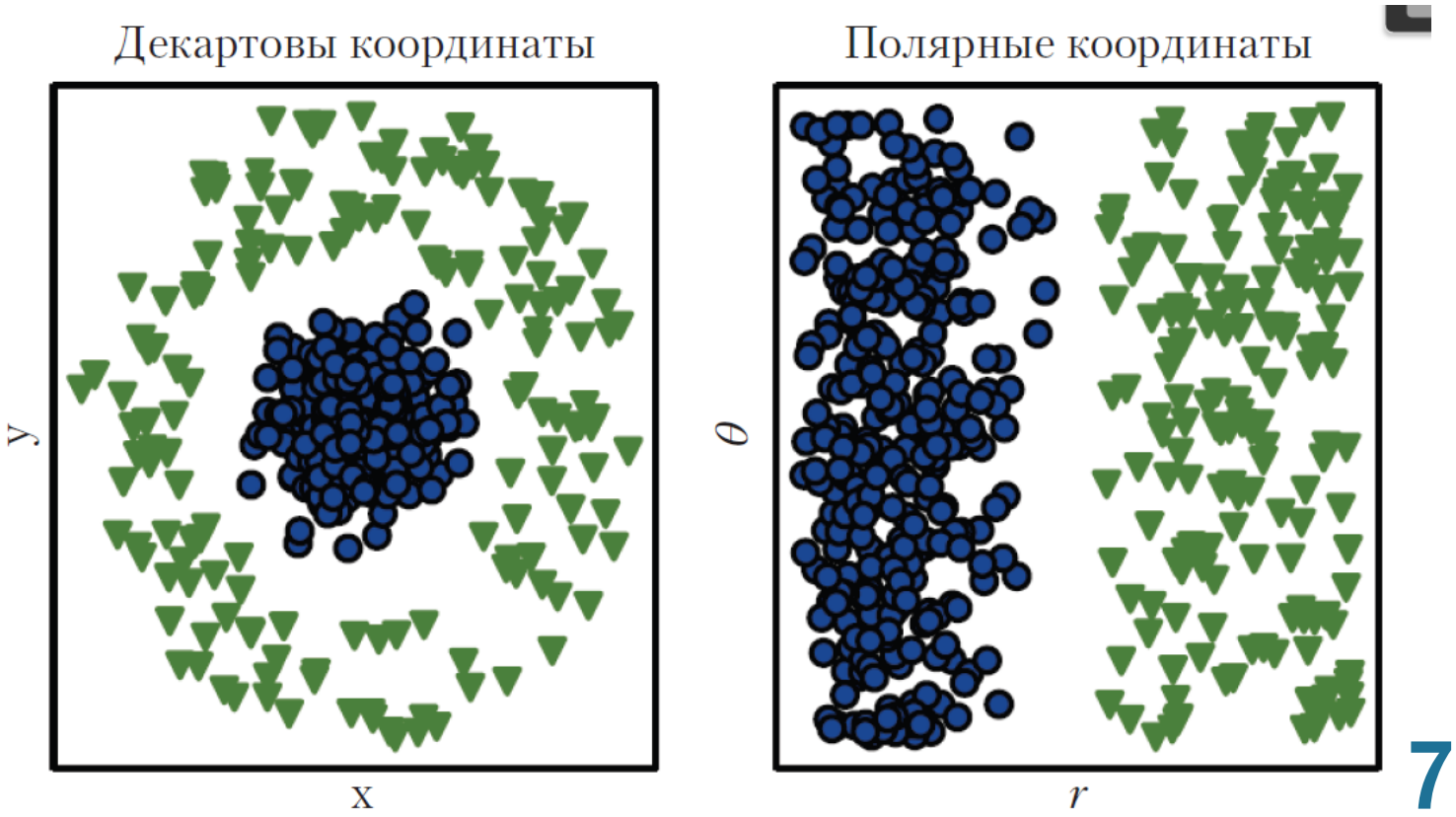
Во многих приложениях требуется изощренная предварительная обработка, потому что исходные данные представлены в виде, **малопригодном для архитектур глубокого обучения**. Пополнение набора данных можно рассматривать как **вид предобработки** одного лишь обучающего набора.

Лекция 1.1 - Сбор данных для глубокого обучения

РАЗНИЦА В ПРЕДСТАВЛЕНИИ РАЗНОРОДНЫХ ДАННЫХ

Можно пользоваться машинным обучением не только для того, чтобы найти отображение представления на результат, но и чтобы определить само представление. Такой подход называется ***обучением представлений***.

Пример различных представлений: предположим, что требуется разделить две категории данных, проведя прямую на диаграмме рассеяния. На левом рисунке данные представлены в декартовых координатах, и задача неразрешима. На правом рисунке те же данные представлены в полярных координатах и разделяются вертикальной прямой.



Лекция 1.1 - Сбор данных для глубокого обучения

ПРЕДСТАВЛЕНИЕ ДАННЫХ

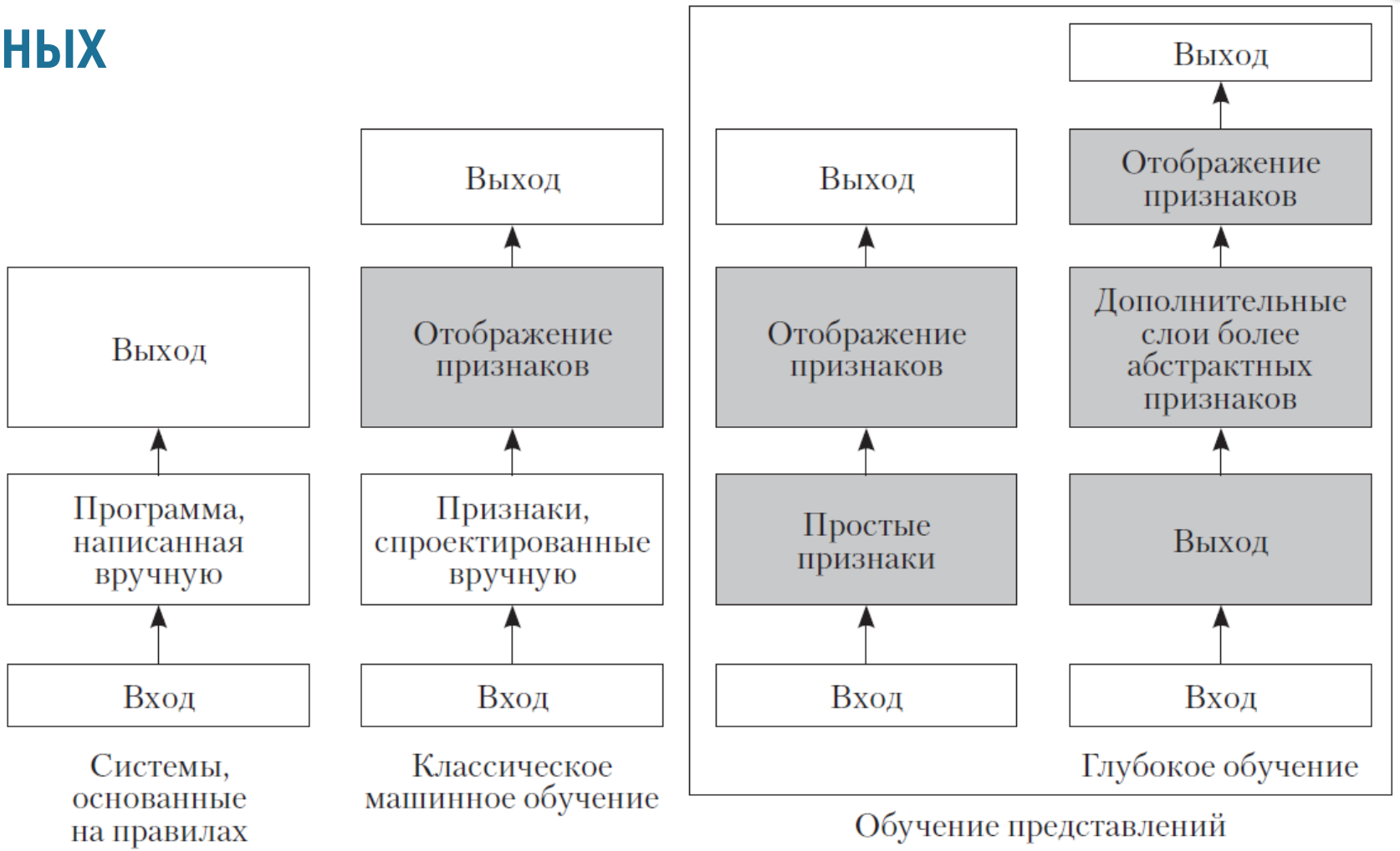
При проектировании признаков или алгоритмов обучения признаков нашей целью обычно является ***выделение факторов вариативности***, которые объясняют ***наблюдаемые данные.*** В этом контексте слово «фактор» означает просто источник влияния, а не «сомножитель». Их можно представлять себе как концепции или абстракции, помогающие извлечь смысл из данных, характеризуемых высокой вариативностью.

Источник трудностей в целом ряде практических приложений искусственного интеллекта – тот факт, что многие факторы вариативности оказывают влияние абсолютно на все данные, доступные нашему наблюдению. Разумеется, может оказаться очень трудно выделить такие высокоуровневые абстрактные признаки из исходных данных.

Глубокое обучение решает эту центральную проблему обучения представлений, вводя представления, выражаемые в терминах других, более простых представлений. Каждое применение одной математической функции можно рассматривать как новое представление входных данных. Идея нахождения подходящего представления данных путем обучения – это лишь один взгляд на глубокое обучение.

Лекция 1.1 - Сбор данных для глубокого обучения

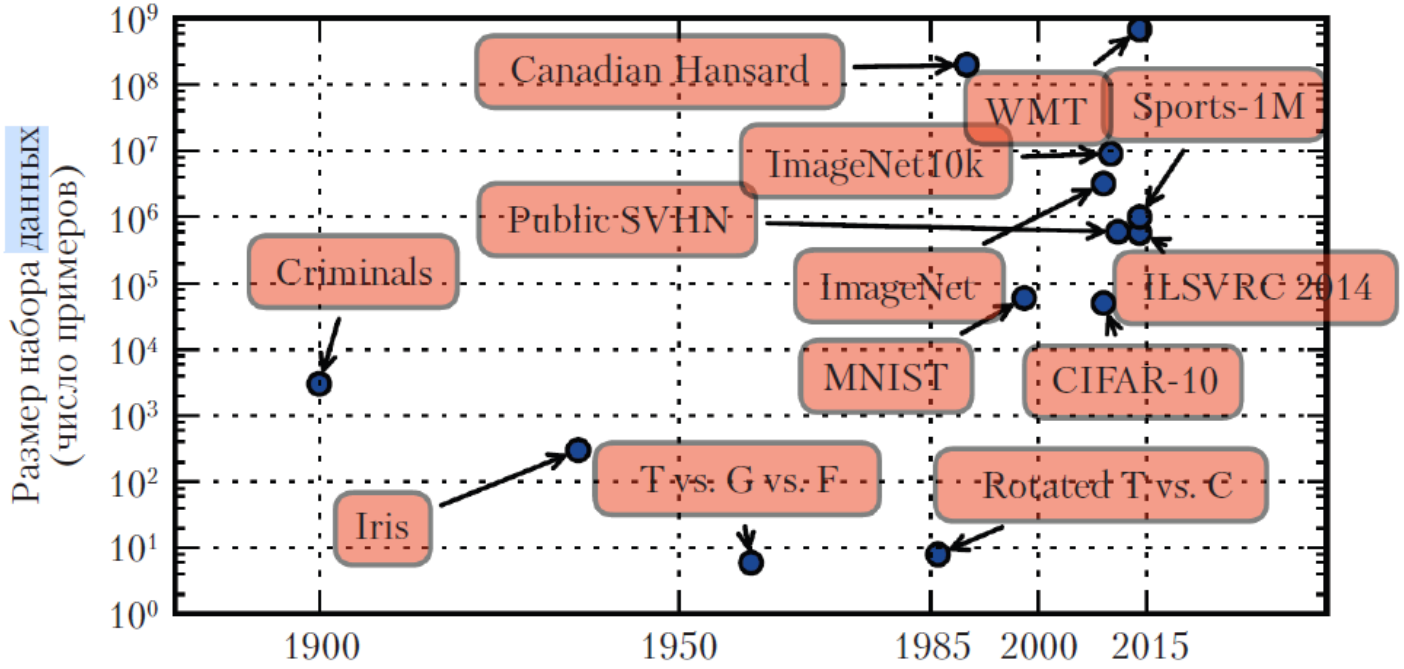
ОБУЧЕНИЕ НА ДАННЫХ



Связь различных частей системы ИИ между собой в рамках разных подходов к ИИ.
Серым цветом показаны компоненты, способные обучаться на данных

Лекция 1.1 - Сбор данных для глубокого обучения

УВЕЛИЧЕНИЕ РАЗМЕРА НАБОРА ДАННЫХ



Увеличение размера набора данных со временем. В начале 1900-х годов статистики изучали наборы данных, содержащие от сотен до тысяч вручную подготовленных измерений. В период между 1950-ми и 1980-ми пионеры биокомпьютерного зрения зачастую работали с небольшими синтетическими наборами данных, например растровыми изображениями букв низкого разрешения, специально спроектированными так, чтобы снизить стоимость вычислений и продемонстрировать, что нейронные сети можно обучить функциям специального вида. В 1980-е и 1990-е машинное обучение стало в большей степени статистическим, а наборы данных уже насчитывали десятки тысяч примеров. В первом десятилетии XXI века продолжали создавать более изощренные наборы данных того же размера, например CIFAR-10. В конце этого периода и в первой половине 2010-х появление гораздо больших наборов данных, содержащих от сотен тысяч до десятков миллионов примеров, полностью изменило представление о возможностях глубокого обучения. К таким наборам относится общедоступный набор номеров домов, различные варианты набора ImageNet и набор Sports-1M. В верхней части диаграммы мы видим, что наборы переведенных предложений, например набор, построенный IBM по официальным отчетам о заседаниях канадского парламента, и набор WMT 2014 переводов с английского на французский, по размеру намного превосходят большинство остальных наборов

Лекция 1.2 - Теоретические основы глубокого обучения

Теория вероятностей – это раздел математики, в котором рассматриваются недостоверные утверждения.

Если теория вероятностей позволяет формулировать недостоверные утверждения и рассуждать в условиях неопределенности, то теория информации дает возможность количественно оценить меру неопределенности распределения вероятности.

Существуют три источника неопределенности:

- ***стохастичность***, присущая моделируемой системе. Например, в большинстве интерпретаций квантовой механики динамика субатомных частиц описывается в вероятностных терминах. Можно также сконструировать теоретические сценарии с постулированной случайной динамикой, например гипотетическая карточная игра в предположении, что карты перетасованы случайно;
- ***неполнота наблюдаемых данных***. Даже детерминированная система может казаться стохастической, если мы не в состоянии наблюдать все переменные, описывающие ее поведение. Например, в парадоксе Монти Холла участник игрового шоу выбирает одну из трех дверей и получает скрытый за ней приз. За двумя дверьми находятся козы, за третьей – автомобиль. Исход при любом выборе участника детерминирован, но, с точки зрения самого участника, исход неопределенный;
- ***неполнота модели***. Если используется модель, которая отбрасывает часть наблюдаемой информации, то отброшенная информация приводит к недостоверности полученных от модели предсказаний. Допустим, к примеру, что мы конструируем робота, который способен точно фиксировать положения всех находящихся поблизости от него объектов. Если робот дискретизирует пространство, стремясь спрогнозировать положения объектов в будущем, то сам акт дискретизации уже делает информацию о положении объектов недостоверной: каждый объект может находиться в дискретной области, окружающей занимаемое им место в пространстве.

Лекция 1.2 - Теоретические основы глубокого обучения

ТЕОРИЯ ВЕРОЯТНОСТЕЙ И ТЕОРИЯ ИНФОРМАЦИИ

Словарь терминов:

- ***Случайной величиной*** называется величина, случайно принимающая различные значения.
- ***Распределение вероятности*** описывает, с какой вероятностью случайная величина или множество случайных величин принимает каждое возможное значение.
- Функция вероятности многих переменных называется ***совместным распределением вероятности***.
- Распределение непрерывных случайных величин описывается ***функцией плотности вероятности***, а не функцией вероятности.
- Иногда известно распределение вероятности множества величин, а мы хотим узнать распределение вероятности подмножества этих величин. Оно называется ***маргинальным распределением вероятности***.
- Вероятность некоторого события, при условии что произошло какое-то другое событие называется ***условной вероятностью***.
- Любое совместное распределение вероятности нескольких случайных величин можно разложить в произведение условных вероятностей одной величины, полученный результат называется ***цепным правилом***, или ***правилом умножения вероятностей.***
- Две случайные величины x и y называются ***независимыми***, если их совместное распределение вероятности можно представить в виде произведения двух сомножителей, один из которых содержит только x, а другой – только y.
- Две случайные величины x и y называются ***условно независимыми*** при условии случайной величины z, если условное распределение вероятности x и y можно представить в виде произведения для любого значения z.
- ***Математическим ожиданием***, или ожидаемым значением, функции f(x) относительно распределения вероятности P(x) называется среднее значение f, когда x выбирается из P.
- ***Дисперсия*** измеряет разброс значений функции случайной величины x с заданным распределением вероятности.
- Если дисперсия мала, то значения f(x) сгруппированы в окрестности ожидаемого значения. Квадратный корень из дисперсии называется ***стандартным отклонением***.
- ***Ковариация*** дает представление о силе линейной связи между двумя функциями случайных величин, а также о масштабе этих величин.

Лекция 1.2 - Теоретические основы глубокого обучения

НАИБОЛЕЕ ЧАСТО ВСТРЕЧАЮЩИЕСЯ РАСПРЕДЕЛЕНИЯ ВЕРОЯТНОСТИ

Распределение Бернулли – это распределение одной случайной величины, принимающей всего два значения. Свойства этого распределения:

$$P(\mathrm{x} = 1) = \phi$$

$$P(\mathrm{x} = 0) = 1 - \phi$$

$$\mathrm{P}(\mathrm{x} = x) = \phi^{x}(1 - \phi)^{1-x}$$

$$\mathbb{E}_{x}[\mathrm{x}] = \phi$$

$$\mathrm{Var}_{x}(\mathrm{x}) = \phi(1 - \phi)$$

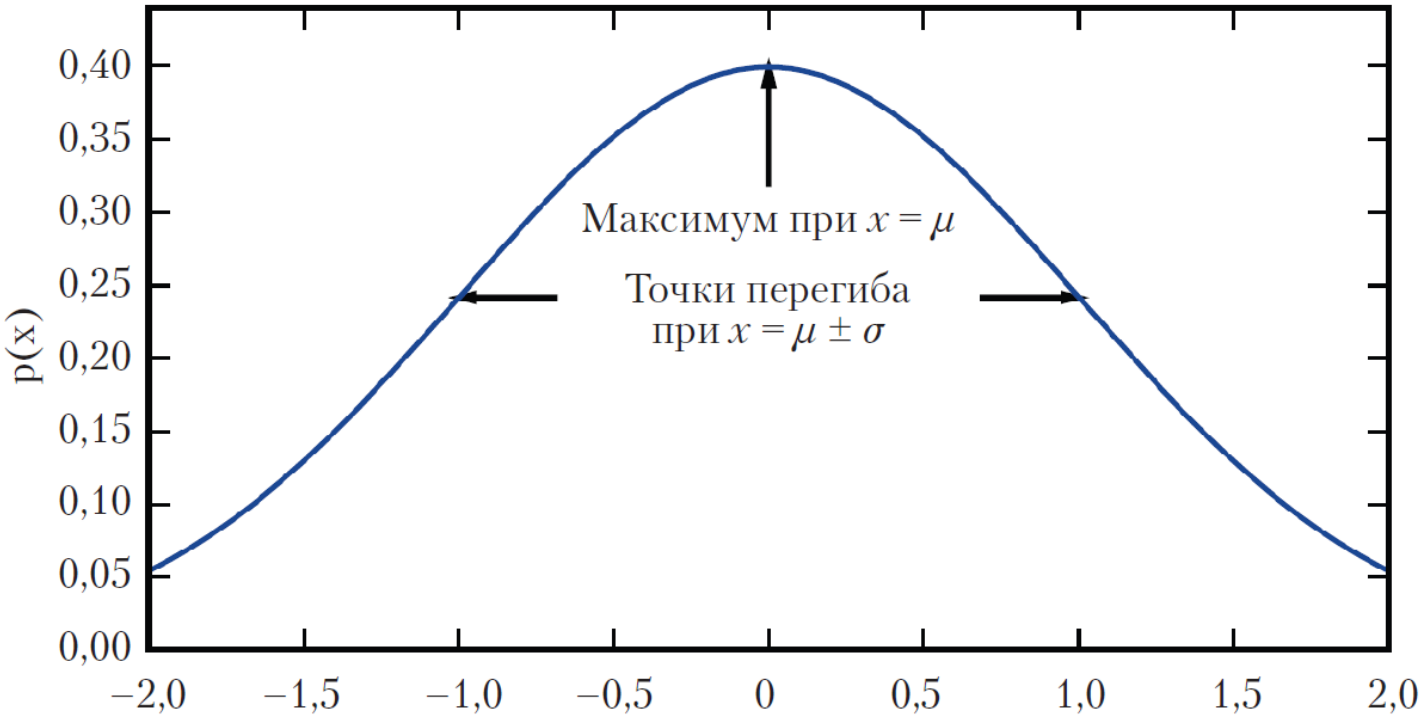
Категориальным распределением (или распределением «multinoulli») называется распределение одной дискретной случайной величины, принимающей конечное число значений k[1]**.** Категориальное распределение параметризовано вектором $\boldsymbol{p} \in [0, 1]^{k-1}$, где p_i – вероятность i-го состояния. Вероятность последнего, k-го, состояния равна $1 - \mathbf{1}^{\top}\mathbf{p}$. При этом необходимо наложить ограничение $\mathbf{1}^{\top}\mathbf{p} \leq 1$. Категориальные распределения часто используются для описания распределения категорий объектов, поэтому мы обычно не будем предполагать, что состоянию 1 соответствует числовое значение 1 и т. д. По этой причине нам, как правило, не нужно вычислять математическое ожидание или дисперсию случайных величин с категориальным распределением.

Лекция 1.2 - Теоретические основы глубокого обучения

НАИБОЛЕЕ ЧАСТО ВСТРЕЧАЮЩИЕСЯ РАСПРЕДЕЛЕНИЯ ВЕРОЯТНОСТИ

Самым распространенным распределением вещественных чисел является ***нормальное***, или ***гауссово***, распределение:

$$\mathcal{N}(x;\mu,\sigma^2)=\sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$



Нормальное распределение $\mathcal{N}(x;\ \mu,\ \sigma^2)$ имеет классическую колоколообразную форму, при этом абсцисса центрального пика задается параметром μ, а ширина пика определяется параметром $\mathcal{N}$. На этом рисунке показано стандартное нормальное распределение с $\mu = 0$, $\sigma = 1$

ЭКСПОНЕНЦИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ И РАСПРЕДЕЛЕНИЕ ЛАПЛАСА

$$\text{Laplace}(x;\mu,\gamma)=\frac{1}{2\gamma}\exp\left(-\frac{|x-\mu|}{\gamma}\right)$$

РАСПРЕДЕЛЕНИЕ ДИРАКА И ЭМПИРИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ

Дельта-функция Дирака:

$$p(x)=\delta(x-\mu)$$

Дельта-распределение Дирака часто применяется в качестве компоненты ***эмпирического распределения***:

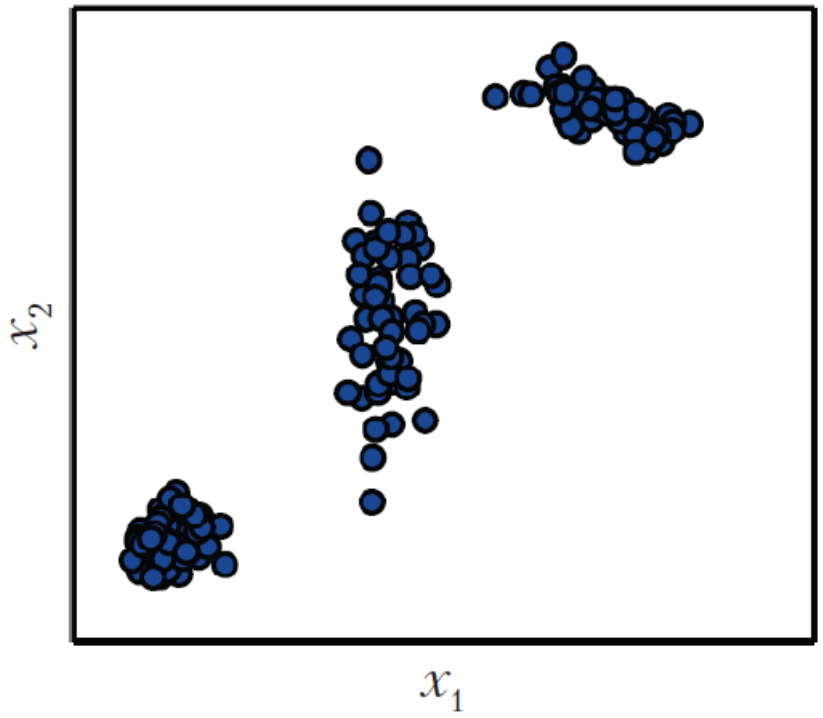
$$\hat{p}(\boldsymbol{x})=\frac{1}{m}\sum_{i=1}^{m}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)})$$

Лекция 1.2 - Теоретические основы глубокого обучения

СМЕСИ РАСПРЕДЕЛЕНИЙ

Один из самых употребительных способов комбинирования – построение ***смеси распределений***, состоящей из нескольких компонент. При каждом испытании определяется, из какого распределения будет производиться выборка, причем для выборки компоненты используется категориальное распределение:

$$P(\mathrm{x}) = \sum_i P(\mathrm{c} = i) P(\mathrm{x} \mid \mathrm{c} = i)$$



Выборки из модели гауссовой смеси. Показаны три компоненты. Если смотреть слева направо, то первая компонента имеет изотропную ковариационную матрицу, т. е. дисперсия во всех направлениях одинакова. Вторая компонента имеет диагональную ковариационную матрицу, т. е. дисперсию можно задавать независимо вдоль каждой оси. В этом примере дисперсия вдоль оси x_2 больше, чем вдоль оси x_1. Третья компонента имеет ковариационную матрицу полного ранга, позволяющую задавать дисперсии вдоль произвольного базиса

Лекция 1.3 - Численные методы для глубокого обучения

ПЕРЕПОЛНЕНИЕ И ПОТЕРЯ ЗНАЧИМОСТИ

Фундаментальная сложность выполнения непрерывных математических операций на цифровом компьютере заключается в том, как представить **бесконечно много вещественных чисел** с помощью **конечного числа комбинаций битов**.

Разновидности ошибок округления:

- ❑ Потеря значимости;
- ❑ Переполнение.

Пример из практики:

Необходимо защищать от потери значимости и переполнения функцию **softmax,** часто применяющуюся для прогнозирования вероятностей, ассоциированных с категориальным распределением.

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

ПЛОХАЯ ОБУСЛОВЛЕННОСТЬ

Под обусловленностью функции понимают скорость ее изменения в ответ на малые изменения аргументов.

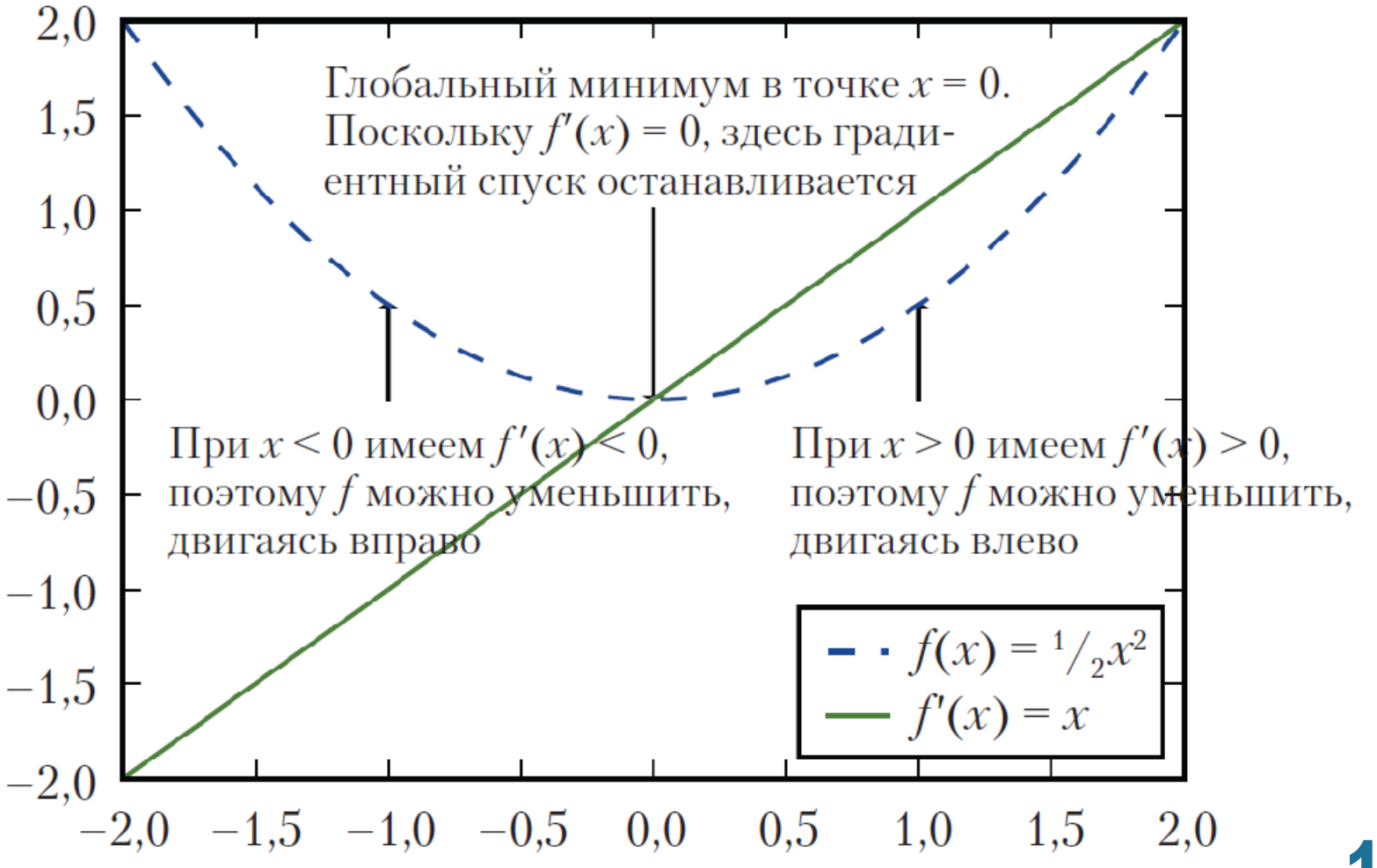
ЧИСЛО ОБУСЛОВЛЕННОСТИ:

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ ГРАДИЕНТНЫМ МЕТОДОМ

Градиентный спуск. Иллюстрация применения метода градиентного спуска с использованием производной для перехода в точку минимума

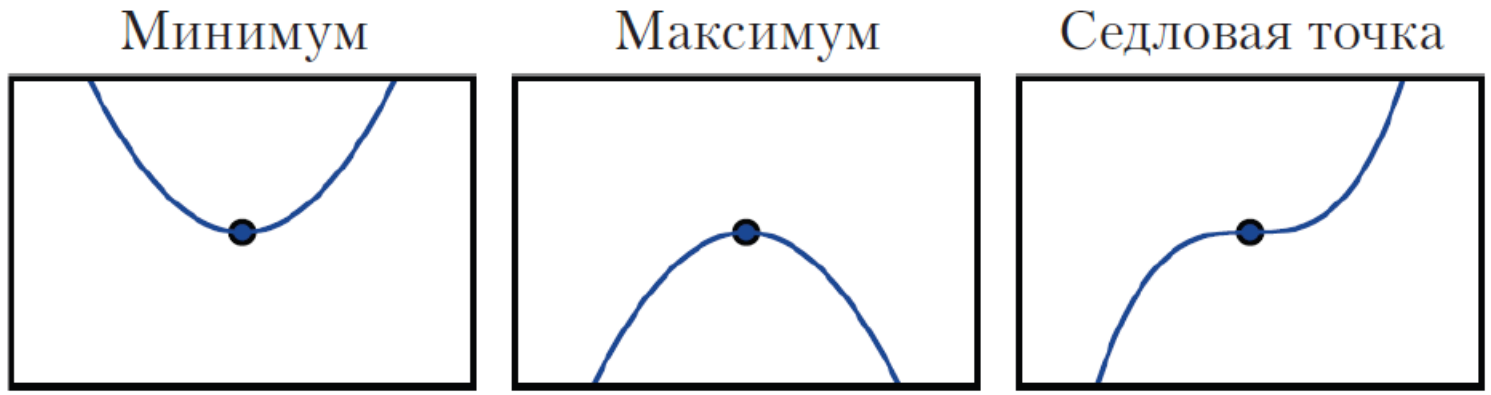


Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ ГРАДИЕНТНЫМ МЕТОДОМ

Критической называется точка с нулевым угловым коэффициентом. Это может быть локальный минимум, в котором значение функции больше значений в окрестных точках, локальный максимум, в котором значение функции меньше значений в окрестных точках, или седловая точка, в которой значение функции может быть как больше, так и меньше значений в окрестных точках.

Типы критических точек. Примеры трех типов критических точек в одномерном случае.



Седловыми называются точки не являющиеся ни локальным максимумом, ни локальным минимумом.

Глобальным минимумом называется точка, в которой достигается абсолютное наименьшее значение функции.

Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ ГРАДИЕНТНЫМ МЕТОДОМ

Приближенная минимизация. Алгоритмы оптимизации могут не найти глобального минимума, если имеется несколько локальных минимумов или плато. В глубоком обучении мы обычно соглашаемся на такие решения, несмотря на то что это не настоящий минимум, при условии что они соответствуют действительно низким значениям функции стоимости

Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ ГРАДИЕНТНЫМ МЕТОДОМ

Производной по направлению в направлении единичного вектора u называется угловой коэффициент функции ***f*** в направлении ***u***. Чтобы уменьшить *f*, мы должны двигаться в направлении отрицательного градиента. Этот алгоритм называется ***методом наискорейшего спуска***, или ***градиентным спуском***. Где предлагается выбирать новую точку (где ***ε*** – скорость обучения):

$$\boldsymbol{x}' = \boldsymbol{x} - \varepsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

Другой подход – вычислить ***f(x – ε∇x f(x))*** для нескольких значений ε и выбрать то, при котором целевая функция принимает наименьшее значение. Эта стратегия называется ***линейным поиском***.

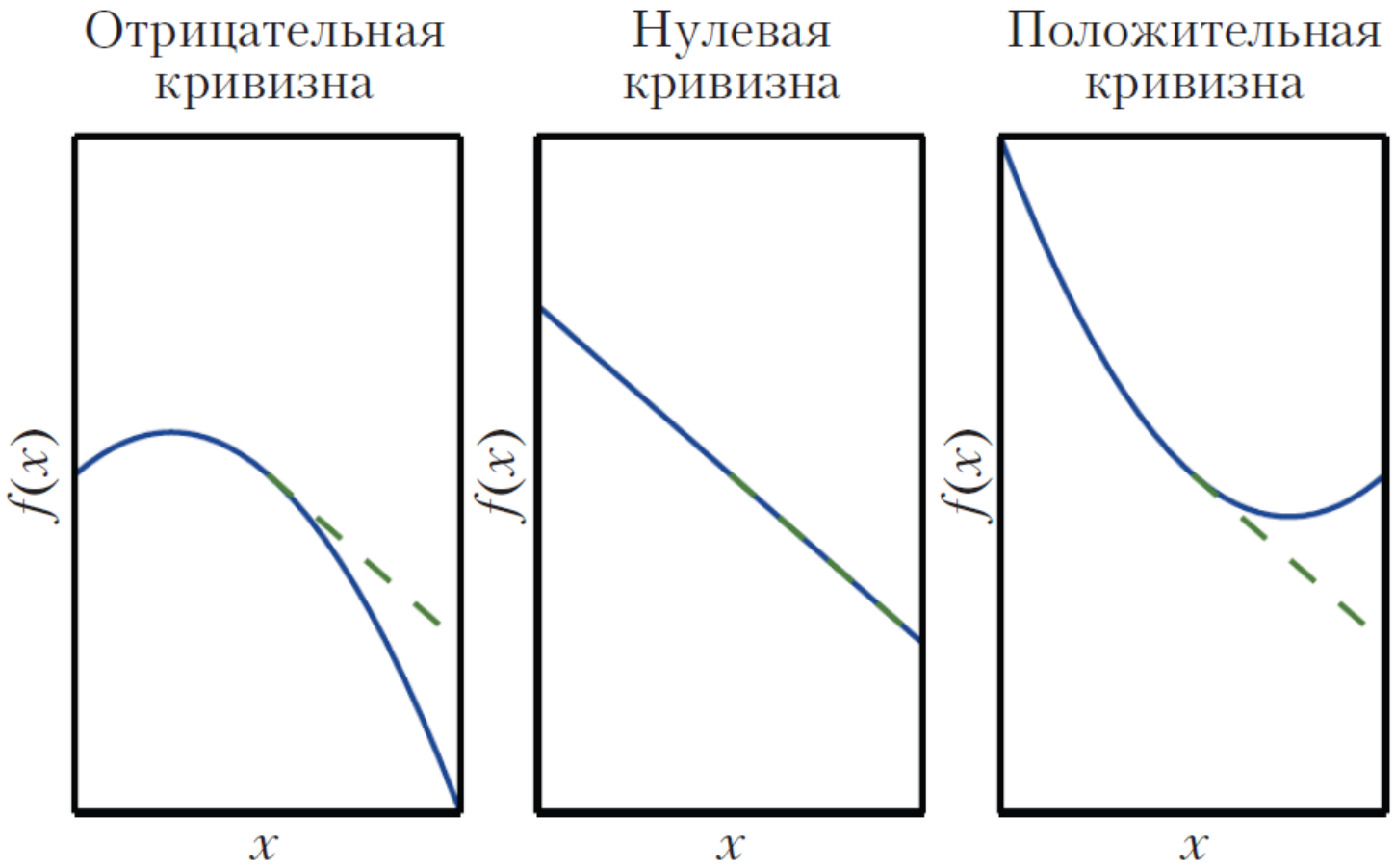
Поиск максимума целевой функции дискретных параметров называется ***методом восхождения на вершину***.

Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ ГРАДИЕНТНЫМ МЕТОДОМ

В случаях, когда требуется найти все частные производные функции, аргументами и значением которой являются векторы. Создается матрица, содержащая все такие производные, которая называется ***матрицей Якоби***, или ***якобианом***.

Вторая производная определяет кривизну функции. Показаны квадратичные функции с разной кривизной. Штриховой линией обозначено значение функции стоимости, ожидаемое на основе анализа одного лишь градиента. Если кривизна отрицательна, то функция стоимости убывает быстрее, чем предсказывает градиент. Если кривизна равна нулю, то градиент правильно предсказывает значение. Если кривизна положительна, то функция стоимости убывает медленнее, чем предсказывает градиент, и в конечном счете начинает возрастать, поэтому если шаг выбран слишком большим, то можно случайно получить завышенное значение

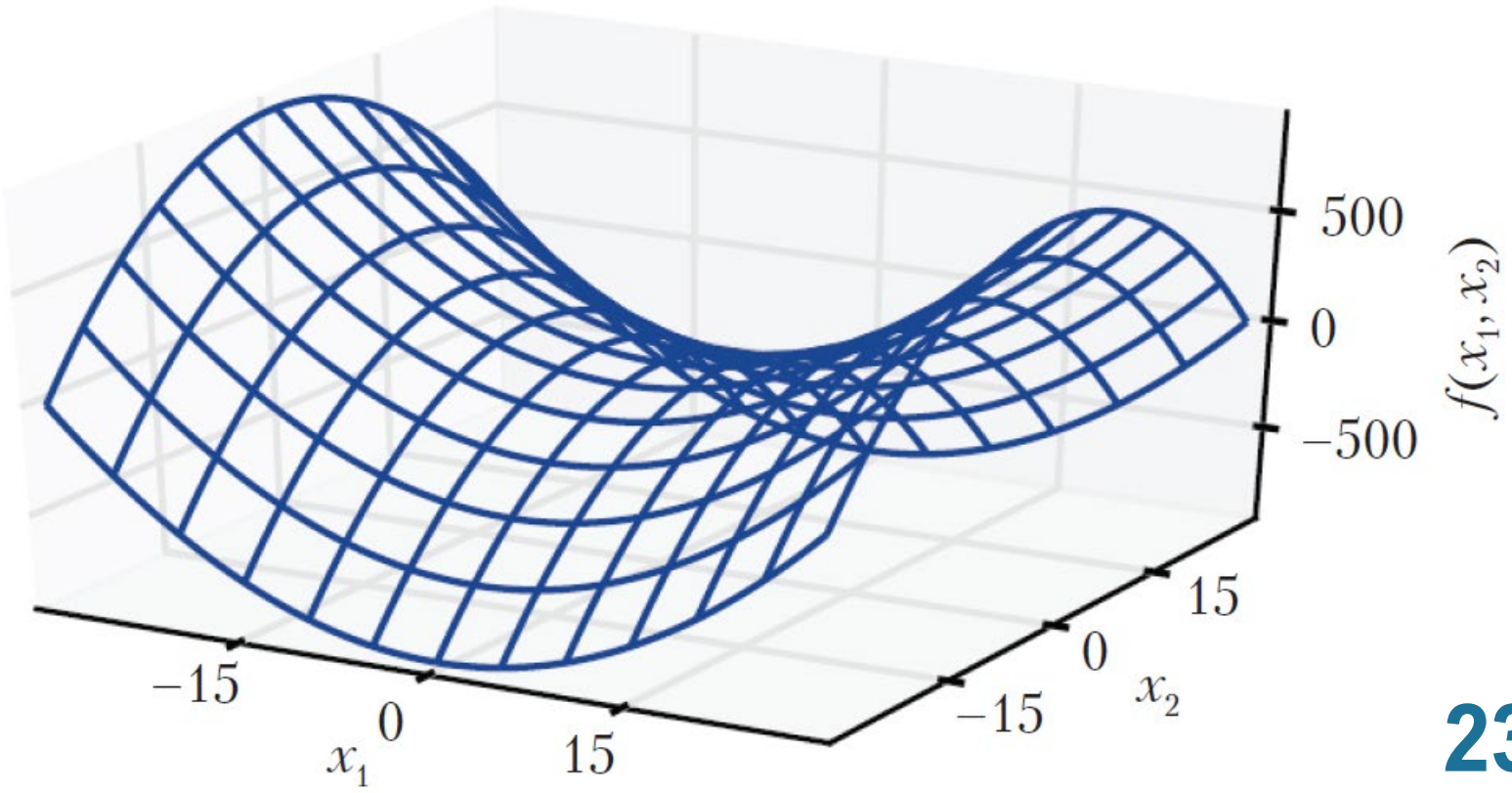


Лекция 1.3 - Численные методы для глубокого обучения

Седловая точка, в которой присутствует как положительная, так и отрицательная кривизна. Представлен график функции $f(x) = x_1^2 - x_2^2$. Вдоль оси x_1 функция изгибается вверх. Направление этой оси – собственный вектор матрицы Гессе с положительным собственным значением. Вдоль оси x_2 функция изгибается вниз. Это направление собственного вектора матрицы Гессе с отрицательным собственным значением. Название «седловая точка» связано с тем, что эта поверхность напоминает седло. Это хрестоматийный пример функции с седловой точкой. В многомерном случае для наличия седловой точки необязательно, чтобы собственное значение было равно 0; необходимо лишь, чтобы существовали как положительные, так и отрицательные собственные значения. В седловой точке, соответствующей собственным значениям разных знаков, в одном сечении достигается локальный максимум, а в другом – локальный минимум.

В случае функции нескольких переменных вторых производных много. Их можно собрать в ***матрицу Гессе***, или ***гессиан***. Матрица Гессе H(f)(x) определяется следующим образом:

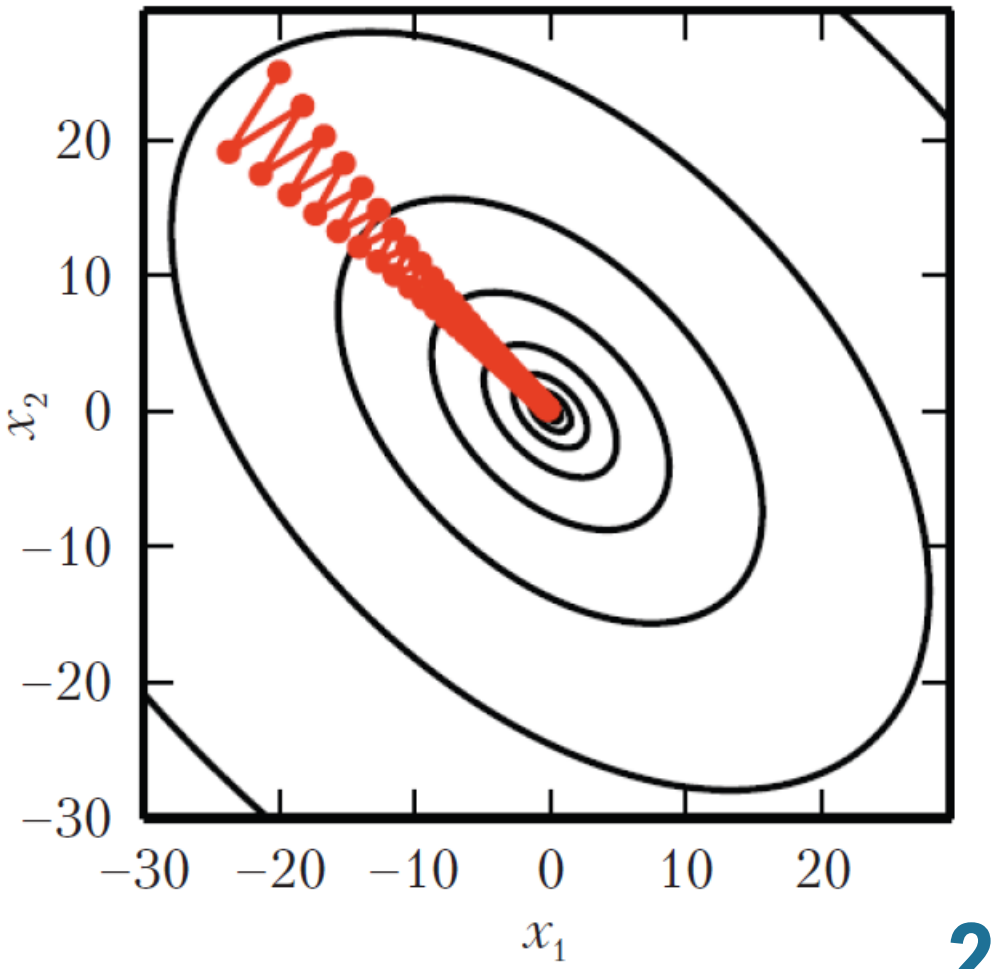
$$\boldsymbol{H}(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$$



Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ ГРАДИЕНТНЫМ МЕТОДОМ

Метод градиентного спуска не использует информацию о кривизне, содержащуюся в гессиане. Здесь градиентный спуск применяется для минимизации квадратичной функции f(x), для которой число обусловленности гессиана равно 5. Это означает, что величина кривизны в направлениях наибольшей и наименьшей кривизны различается в пять раз. В данном случае направление наибольшей кривизны совпадает с вектором $[1, 1]^\top$, а наименьшей – с вектором $[1, -1]^\top$. Красной линией показан путь, которым следует метод градиентного спуска. Эта вытянутая квадратичная функция напоминает длинный каньон. Градиентный спуск бесполезно тратит много времени, раз за разом спускаясь по стенкам каньона, поскольку это направления наискорейшего спуска. Так как шаг слишком большой, мы проскакиваем мимо дна каньона и вынуждены спускаться по противоположной стенке на следующей итерации. Большое положительное значение матрицы Гессе, соответствующее собственному вектору в этом направлении, подсказывает, что производная по этому направлению быстро возрастает, поэтому алгоритм оптимизации мог бы воспользоваться гессианом и понять, что в данном случае производить поиск в направлении наискорейшего спуска не стоит.



Лекция 1.3 - Численные методы для глубокого обучения

ОПТИМИЗАЦИЯ С ОГРАНИЧЕНИЯМИ

Иногда задача состоит в том, чтобы найти максимум или минимум функции f(x) не во всей области допустимых значений x, а лишь в некотором ее подмножестве $\mathbb{S}$. Такая задача называется ***оптимизацией с ограничениями***, или ***ограниченной оптимизацией***. В этом случае точки множества $\mathbb{S}$ называются ***допустимыми***.

Метод Каруша–Куна–Таккера предлагает очень общий подход к решению задач оптимизации. Вводится новая функция, называемая ***обобщенным лагранжианом***, или ***обобщенной функцией Лагранжа***.

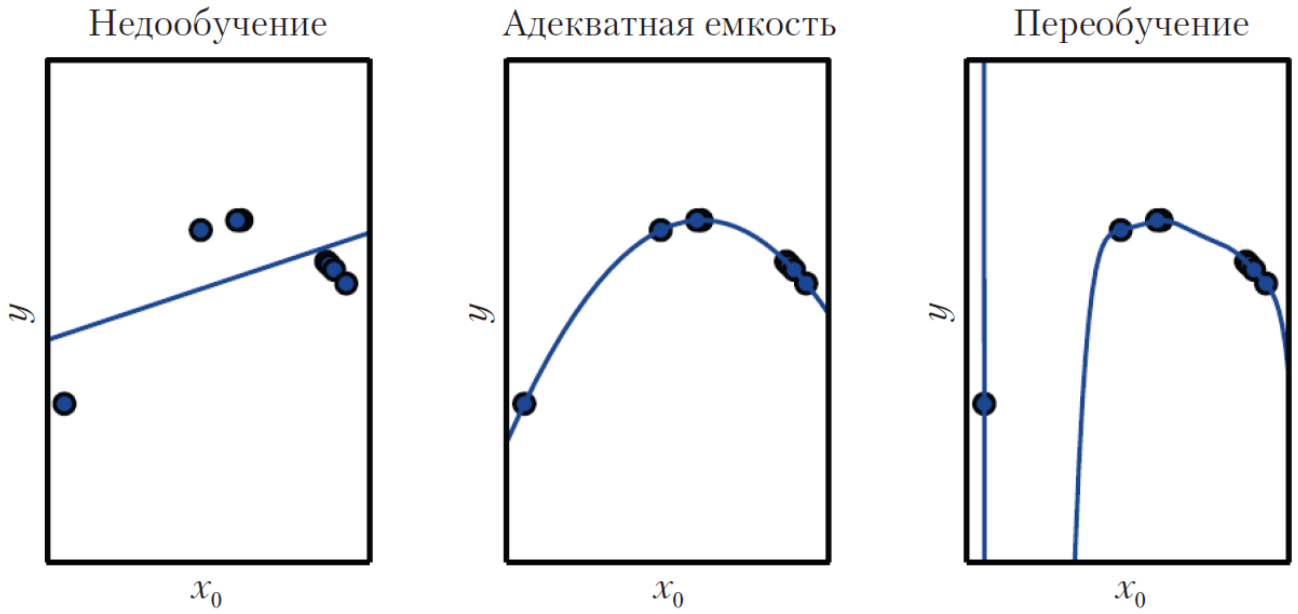
ПРИМЕР: ЛИНЕЙНЫЙ МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

Алгоритм минимизации $f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{Ax} - \boldsymbol{b}||_2^2$ относительно x методом градиентного спуска, начиная с произвольного значения $\boldsymbol{x}$

Взять в качестве величины шага (ε) и допуска (δ) небольшие положительные числа.
while $||\boldsymbol{A}^\top\boldsymbol{Ax} - \boldsymbol{A}^\top\boldsymbol{b}||_2 > \delta$ **do**
    $\boldsymbol{x} \leftarrow \boldsymbol{x} - \varepsilon(\boldsymbol{A}^\top\boldsymbol{Ax} - \boldsymbol{A}^\top\boldsymbol{b})$
end while

Лекция 1.4 - Гиперпараметры, переобучение и недообучение

ГИПЕРПАРАМЕТРЫ

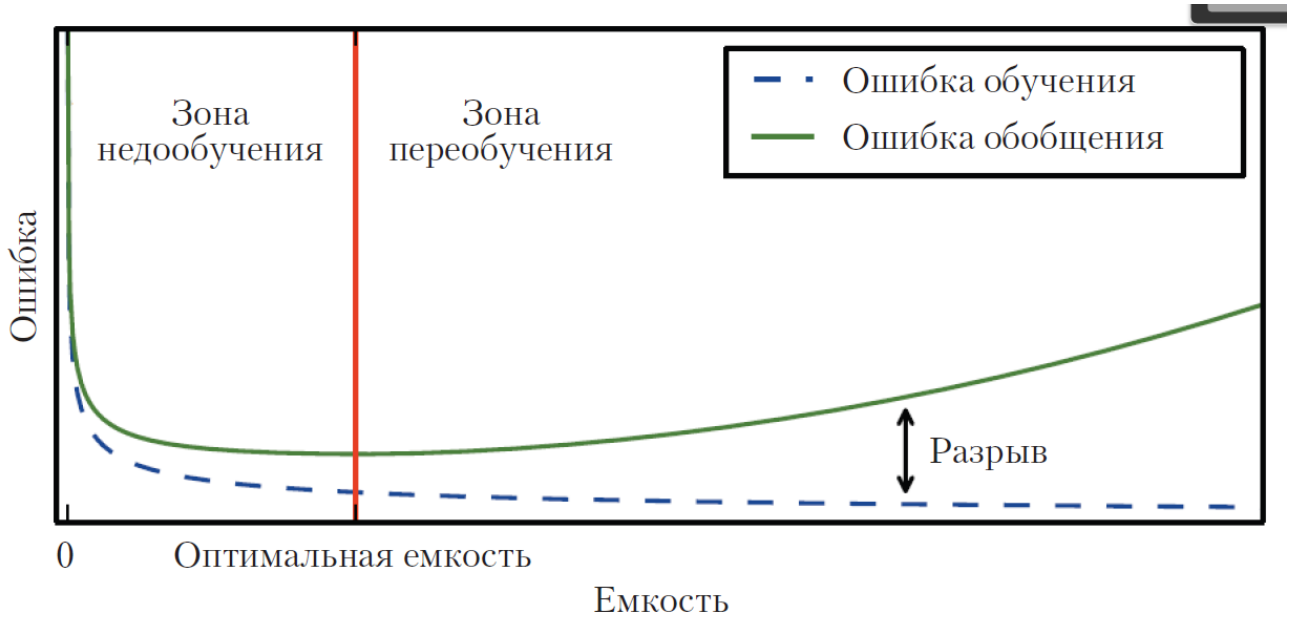


На одном обучающем наборе обучены три модели. Обучающие данные сгенерированы синтетически: значения x выбирались случайно, а значения y вычислялись детерминированно путем применения квадратичной функции. (Слева) Линейная функция, аппроксимирующая данные, страдает от недообучения – она не улавливает присущую данным кривизну. (В центре) Квадратичная функция, аппроксимирующая данные, хорошо обобщается на новые точки. Ей не свойственны ни недообучение, ни переобучение. (Справа) Многочлен степени 9 аппроксимирует данные, но страдает от переобучения. Для решения недоопределенной системы нормальных уравнений мы воспользовались псевдообратной матрицей Мура–Пенроуза. Найденная кривая проходит через все обучающие точки, но мы не смогли выявить правильную структуру. Между двумя обучающими точками присутствует глубокая впадина, которой нет в истинной функции. Кроме того, функция резко возрастает слева от данных, тогда как истинная функция в этой области убывает

Лекция 1.4 - Гиперпараметры, переобучение и недообучение

ГИПЕРПАРАМЕТРЫ

У большинства алгоритмов машинного обучения имеются гиперпараметры, управляющие поведением алгоритма. Значения гиперпараметров не отыскиваются самим алгоритмом (хотя можно построить вложенную процедуру, в которой один алгоритм обучения будет находить гиперпараметры для другого).



В примере полиномиальной регрессии на рисунке прошлого слайда один гиперпараметр: степень многочлена, он играет роль гиперпараметра емкости. Другой пример гиперпараметра – значение λ, контролирующее силу снижения весов. Иногда настройку делают гиперпараметром, а не обучают, потому что оптимизация слишком сложна. Но чаще причина в том, что бессмысленно обучать гиперпараметр на обучающем наборе. Это относится ко всем гиперпараметрам, управляющим емкостью модели. При попытке обучить их на обучающем наборе всегда выбиралась бы максимально возможная емкость модели, что приводило бы к переобучению (рисунок текущего слайда). Например, мы всегда можем взять многочлен высокой степени и положить коэффициент снижения весов $\lambda = 0$ – аппроксимация обучающего набора при этом будет лучше, чем для многочлена более низкой степени и положительного λ. Для решения этой проблемы нам нужен контрольный набор примеров, которых алгоритм не видел в процессе обучения.

Лекция 1.4 - Гиперпараметры, переобучение и недообучение

ЕМКОСТЬ, ПЕРЕОБУЧЕНИЕ И НЕДООБУЧЕНИЕ

Главная проблема машинного обучения состоит в том, что алгоритм должен хорошо работать на новых данных, которых он раньше не видел, а не только на тех, что использовались для обучения модели. Эта способность правильной работы на ранее не предъявлявшихся данных называется ***обобщением***.

Обычно при обучении модели мы имеем доступ к обучающему набору: можем вычислить некоторую меру ошибки на обучающем наборе, которая называется ***ошибкой обучения***, и постараться минимизировать ее. То, что мы только что описали, – всего лишь задача оптимизации. Машинное обучение отличается от оптимизации тем, что мы хотим еще и уменьшить ***ошибку обобщения*** (ее также называют ***ошибкой тестирования***).

Факторов, определяющих качество работы алгоритма машинного обучения, два:

- ❑ сделать ошибку обучения как можно меньше;
- ❑ сократить разрыв между ошибками обучения и тестирования.

Эти факторы соответствуют двум центральным проблемам машинного обучения: недообучению и переобучению. Недообучение имеет место, когда модель не позволяет получить достаточно малую ошибку на обучающем наборе, а переобучение – когда разрыв между ошибками обучения и тестирования слишком велик. Управлять склонностью модели к переобучению или недообучению позволяет ее ***емкость*** (capacity).

Лекция 1.4 - Гиперпараметры, переобучение и недообучение

ЕМКОСТЬ, ПЕРЕОБУЧЕНИЕ И НЕДООБУЧЕНИЕ

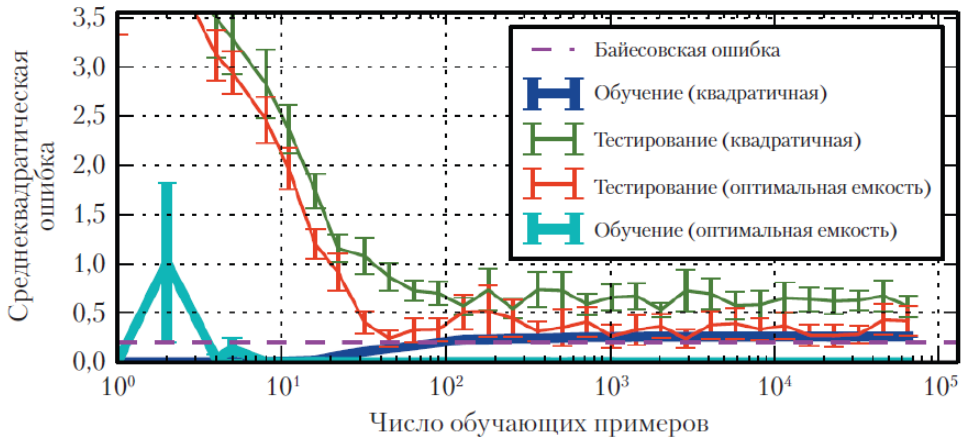
Один из способов контроля над емкостью алгоритма обучения состоит в том, чтобы выбрать его ***пространство гипотез*** – множество функций, которые алгоритм может рассматривать в качестве потенциального решения.

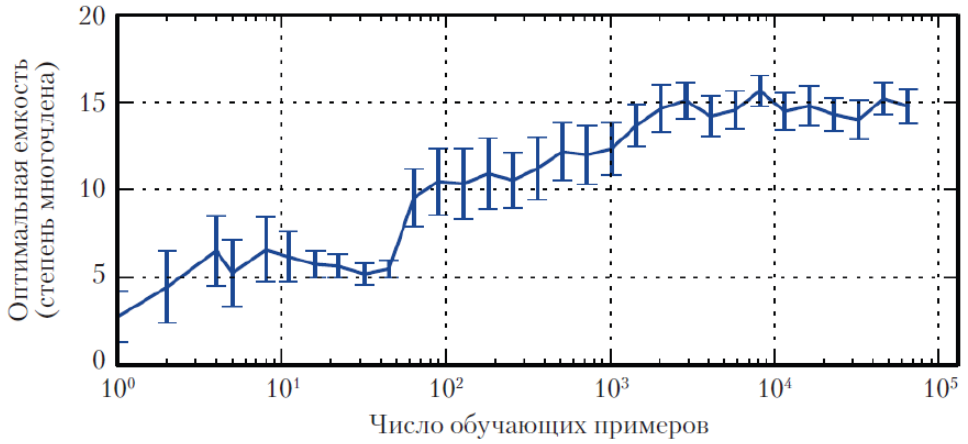
Модель задает семейство функций, из которого может выбирать алгоритм обучения в процессе варьирования параметров. Это называется ***репрезентативной емкостью*** модели. Во многих случаях отыскание наилучшей функции в семействе является трудной задачей оптимизации. На практике алгоритм обучения находит не лучшую функцию, а лишь такую, которая существенно уменьшает ошибку обучения. Наличие дополнительных ограничений, в частности несовершенство алгоритма оптимизации, означает, что ***эффективная емкость*** алгоритма обучения может быть меньше репрезентативной емкости модели.

Аппроксимация обучающего набора, полиномиальной моделью регрессии высокой степени.

Лекция 1.4 - Гиперпараметры, переобучение и недообучение

ЕМКОСТЬ, ПЕРЕОБУЧЕНИЕ И НЕДООБУЧЕНИЕ





Влияние размера обучающего набора данных на ошибки обучения и тестирования, а также на оптимальную емкость модели. Мы синтезировали задачу регрессии, добавив умеренный шум к многочлену степени 5, сгенерировали один тестовый набор, а затем несколько обучающих наборов разного размера. Для каждого размера было сгенерировано 40 различных обучающих наборов, чтобы нанести на график отрезки, отражающие 95%-ные доверительные интервалы. (Вверху) Среднеквадратическая ошибка на обучающем и тестовом наборах для двух разных моделей: квадратичной и полиномиальной, для которой выбрана степень, минимизирующая тестовую ошибку. Обе модели выражены в замкнутой форме. Для квадратичной модели ошибка обучения возрастает с ростом обучающего набора, поскольку чем больше набор, тем труднее его аппроксимировать. Одновременно ошибка тестирования убывает, поскольку меньше неправильных гипотез совместимо с обучающими данными. Емкость квадратичной модели недостаточна для решения этой задачи, поэтому ошибка тестирования асимптотически приближается к высокому значению. Ошибка тестирования при оптимальной емкости асимптотически приближается к байесовской ошибке. Ошибка обучения может стать ниже байесовской, поскольку алгоритм обучения способен запоминать конкретные экземпляры обучающего набора. Когда размер обучающего набора стремится к бесконечности, ошибка обучения любой модели фиксированной емкости (в данном случае квадратичной) должна возрастать как минимум до байесовской ошибки. (Внизу) С ростом размера обучающего набора оптимальная емкость (показанная здесь как степень оптимального полиномиального регрессора) увеличивается. Оптимальная емкость выходит на плато после достижения сложности, достаточной для решения задачи.

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

Такие фундаментальные понятия, как оценивание параметров, смещение и дисперсия, полезны для формальной характеристики обобщения, недообучения и переобучения.

ТОЧЕЧНОЕ ОЦЕНИВАНИЕ

Алгоритм k-групповой перекрестной проверки. Применяется для оценивания ошибки обобщения алгоритма обучения A, когда имеющийся набор данных $\mathbb{D}$ слишком мал для того, чтобы простое разделение на обучающий и тестовый или обучающий и контрольный наборы могло дать точную оценку ошибки обобщения, поскольку среднее значение потери L на малом тестовом наборе может иметь высокую дисперсию.

Define `KFoldXV`($\mathbb{D}$, A, L, k):
Require: $\mathbb{D}$ – набор данных, состоящий из элементов $\boldsymbol{z}^{(i)}$
Require: A – алгоритм обучения, т. е. функция, которая принимает набор данных и возвращает обученную функцию
Require: L – функция потерь, т. е. функция от обученной функции f и примера $\boldsymbol{z}^{(i)} \in \mathbb{D}$, возвращающая скаляр $\in \mathbb{R}$
Require: k – число групп
  Разбить $\mathbb{D}$ на k непересекающихся подмножеств $\mathbb{D}_i$, объединение которых равно $\mathbb{D}$
  for i от 1 до k **do**
    $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$
    for $\boldsymbol{z}^{(j)}$ из $\mathbb{D}_i$ **do**
      $e_j = L(f_i, \boldsymbol{z}^{(j)})$
    end for
  end for
  Return $\boldsymbol{e}$

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

СМЕЩЕНИЕ

$$\text{bias}(\hat{\boldsymbol{\theta}}_m) = \mathbb{E}(\hat{\boldsymbol{\theta}}_m) - \boldsymbol{\theta}$$

где математическое ожидание вычисляется по данным (рассматриваемым как выборка из случайной величины), а θ – истинное значение параметра, которое определяет порождающее распределение. Оценка θˆm называется несмещенной, если bias(θˆm) = 0, т. е. $\mathbb{E}$(θˆm) = θ. Оценка θˆm называется асимптотически несмещенной, если limm⟶∞ bias(θˆm) = 0, т. е. limm⟶∞ $\mathbb{E}$(θˆm) = θ.

Пример: оценка среднего нормального распределения. Рассмотрим множество независимых примеров {x(1), …, x(m)}, имеющих одно и то же нормальное распределение p(x(i)) = $\mathcal{N}$(x(i), μ; σ2), где i ∈ {1, …, m}.

Стандартная оценка среднего нормального распределения называется выборочным средним:

$$\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

Чтобы найти смещение выборочного среднего, нужно вычислить его математическое ожидание:

$$\begin{aligned}
\text{bias}(\hat{\mu}_m) &= \mathbb{E}[\hat{\mu}_m] - \mu \\
&= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right] - \mu \\
&= \left(\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}[x^{(i)}]\right) - \mu \\
&= \left(\frac{1}{m}\sum_{i=1}^{m} \mu\right) - \mu \\
&= \mu - \mu = 0
\end{aligned}$$

Таким образом, выборочное среднее – несмещенная оценка среднего значения нормального распределения

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

ДИСПЕРСИЯ И СТАНДАРТНАЯ ОШИБКА

Дисперсией оценки называется выражение: Var(θˆ) где случайной величиной является обучающий набор. Стандартной ошибкой SE(θˆ) называется квадратный корень из дисперсии. Стандартная ошибка среднего равна:

$$\mathrm{SE}(\hat{\mu}_m) = \sqrt{\mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}}$$

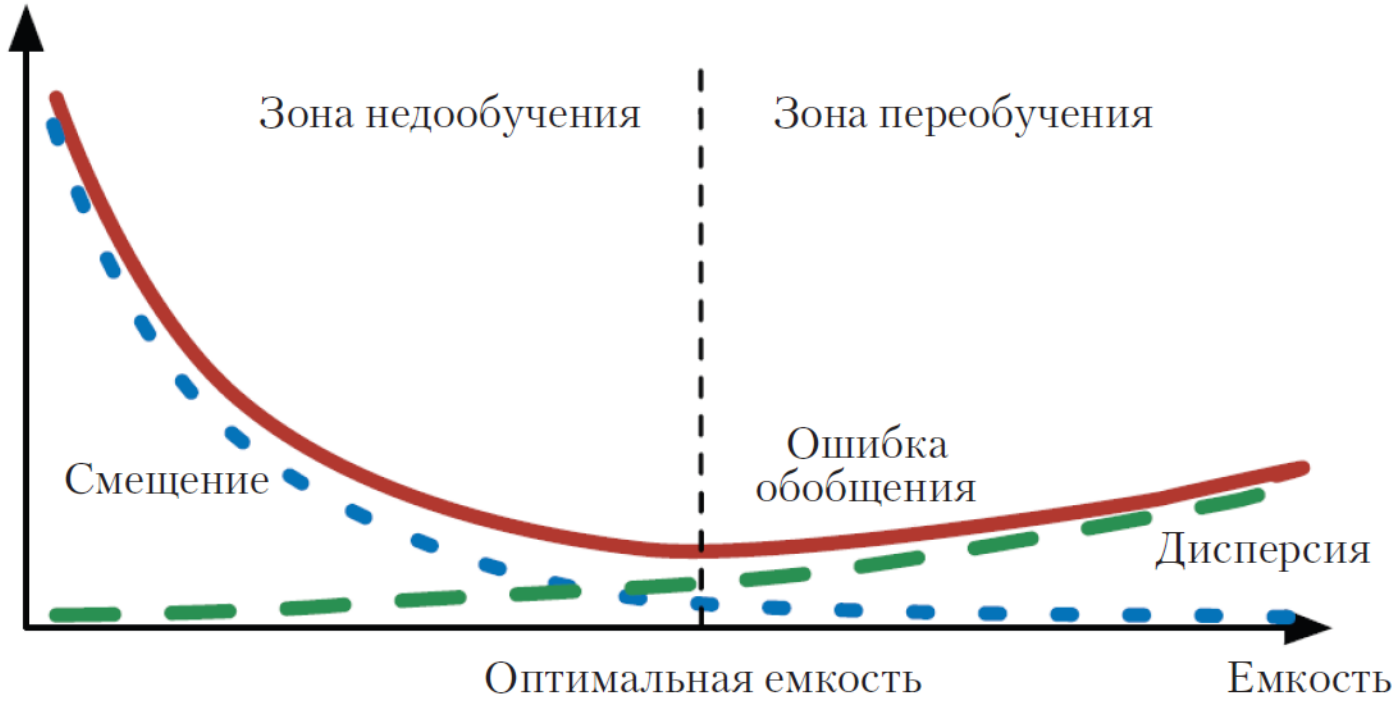
Пример: распределение Бернулли. Снова рассмотрим множество независимых примеров $\{x^{(1)}, \ldots, x^{(m)}\}$, имеющих одно и то же распределение Бернулли.

$$\begin{aligned}
\mathrm{Var}(\hat{\theta}_m) &= \mathrm{Var}\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right) \\
&= \frac{1}{m^2}\sum_{i=1}^{m} \mathrm{Var}(x^{(i)}) \\
&= \frac{1}{m^2}\sum_{i=1}^{m} \theta(1-\theta) \\
&= \frac{1}{m^2} m\theta(1-\theta) \\
&= \frac{1}{m}\theta(1-\theta)
\end{aligned}$$

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

ПОИСК КОМПРОМИССА МЕЖДУ СМЕЩЕНИЕМ И ДИСПЕРСИЕЙ ДЛЯ МИНИМИЗАЦИИ СРЕДНЕКВАДРАТИЧЕСКОЙ ОШИБКИ

С ростом емкости (ось x) смещение (пунктирная линия) уменьшается, а дисперсия (штриховая линия) увеличивается, в результате чего получается еще одна U-образная кривая для ошибки обобщения (жирная линия). В какой-то точке на оси x емкость оптимальна, слева от этой точки имеет место недообучение, справа – переобучение.



Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

БАЙЕСОВСКАЯ СТАТИСТИКА

До сих пор мы обсуждали частотные статистики и подходы, основанные на оценивании единственного значения θ, которое затем использовалось для всех предсказаний. Другой подход состоит в том, чтобы делать предсказание, рассматривая все возможные значения θ. Это область ***байесовской статистики***.

Вероятность в этом случае отражает степень уверенности в наших знаниях. Набор данных доступен прямому наблюдению и, следовательно, не является случайным. С другой стороны, истинный параметр θ неизвестен или недостоверен и потому представляется случайной переменной.

Рассмотрим теперь набор примеров $\{x^{(1)}, \ldots, x^{(m)}\}$. Можно реконструировать влияние данных на наши гипотезы про θ, объединив правдоподобие данных $p(x^{(1)}, \ldots, x^{(m)} \mid \theta)$ с априорным распределением посредством правила Байеса:

$$p(\theta \mid x^{(1)}, \ldots, x^{(m)}) = \frac{p(x^{(1)}, \ldots, x^{(m)} \mid \theta) p(\theta)}{p(x^{(1)}, \ldots, x^{(m)})}$$

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

БАЙЕСОВСКАЯ СТАТИСТИКА

Пример: байесовская линейная регрессия. Рассмотрим байесовский подход к обучению параметров линейной регрессии. Мы хотим обучить линейное отображение входного вектора $x \in \mathbb{R}^n$ для предсказания скалярного значения $y \in \mathbb{R}$. Предсказание параметризуется вектором $w \in \mathbb{R}^n$:

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$$

Если дано множество m обучающих примеров ($X^{(train)}$, $y^{(train)}$), то предсказание y по всему обучающему набору можно выразить в виде:

$$\hat{\boldsymbol{y}}^{(\text{train})} = \boldsymbol{X}^{(\text{train})}\boldsymbol{w}$$

Предполагая нормальное условное распределение $y^{(train)}$, имеем:

$$p(\boldsymbol{y}^{(\text{train})} \mid \boldsymbol{X}^{(\text{train})}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}^{(\text{train})}; \boldsymbol{X}^{(\text{train})}\boldsymbol{w}, \boldsymbol{I}),$$

$$\propto \exp(-{}^1/_2(\boldsymbol{y}^{(\text{train})} - \boldsymbol{X}^{(\text{train})}\boldsymbol{w})^\top(\boldsymbol{y}^{(\text{train})} - \boldsymbol{X}^{(\text{train})}\boldsymbol{w}))$$

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

БАЙЕСОВСКАЯ СТАТИСТИКА

Для определения апостериорного распределения вектора параметров модели w нужно сначала задать априорное распределение. Оно должно отражать наши наивные представления о ценности параметров. Иногда выразить априорные представления в терминах параметров модели трудно или неестественно, но на практике мы обычно предполагаем довольно широкое распределение, выражающее высокую степень неопределенности θ. Для вещественных параметров часто в качестве априорного берут нормальное распределение.

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0) \propto \exp(-{}^1/_2(\boldsymbol{w} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0^{-1}(\boldsymbol{w} - \boldsymbol{\mu}_0))$$

При таком задании априорного распределения мы теперь можем перейти к определению апостериорного распределения параметров модели:

$$\begin{aligned} p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) &\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w}), \\ &\propto \exp(-{}^1/_2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})) \exp(-{}^1/_2(\boldsymbol{w} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0^{-1}(\boldsymbol{w} - \boldsymbol{\mu}_0)) \\ &\propto \exp(-{}^1/_2(-2\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{\Lambda}_0^{-1}\boldsymbol{w} - 2\boldsymbol{\mu}_0^\top \boldsymbol{\Lambda}_0^{-1}\boldsymbol{w})). \end{aligned}$$

Лекция 1.5 - Оценки смещения и байесовские статистики в глубоком обучении

БАЙЕСОВСКАЯ СТАТИСТИКА

При переопределении новых переменных апостериорное распределение можно записать в виде нормального:

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) \propto \exp(-{}^1/_2(\boldsymbol{w} - \boldsymbol{\mu}_m)^\top \boldsymbol{\Lambda}_m^{-1}(\boldsymbol{w} - \boldsymbol{\mu}_m) + {}^1/_2\boldsymbol{\mu}_m^\top \boldsymbol{\Lambda}_m^{-1}\boldsymbol{\mu}_m),$$

$$\propto \exp(-{}^1/_2(\boldsymbol{w} - \boldsymbol{\mu}_m)^\top \boldsymbol{\Lambda}_m^{-1}(\boldsymbol{w} - \boldsymbol{\mu}_m)).$$

Изучение этого апостериорного распределения позволяет составить интуитивное представление о поведении байесовского вывода. В большинстве случаев мы задаем μ_0 равным 0. Если положить $\Lambda_0 = (1/\alpha)I$, то μ_m дает ту же оценку w, что и частотная линейная регрессия со снижением веса $\alpha w^\top w$. Одно отличие заключается в том, что байесовская оценка не определена, если α равно 0 – запрещается начинать процесс байесовского обучения с бесконечно широким априорным распределением w. Но есть и более важное отличие – байесовская оценка дает ковариационную матрицу, показывающую, насколько вероятны все значения w, а не только оценку μ_m.

Лекция 1.6 - Задачи искусственного интеллекта, требующие глубокого обучения

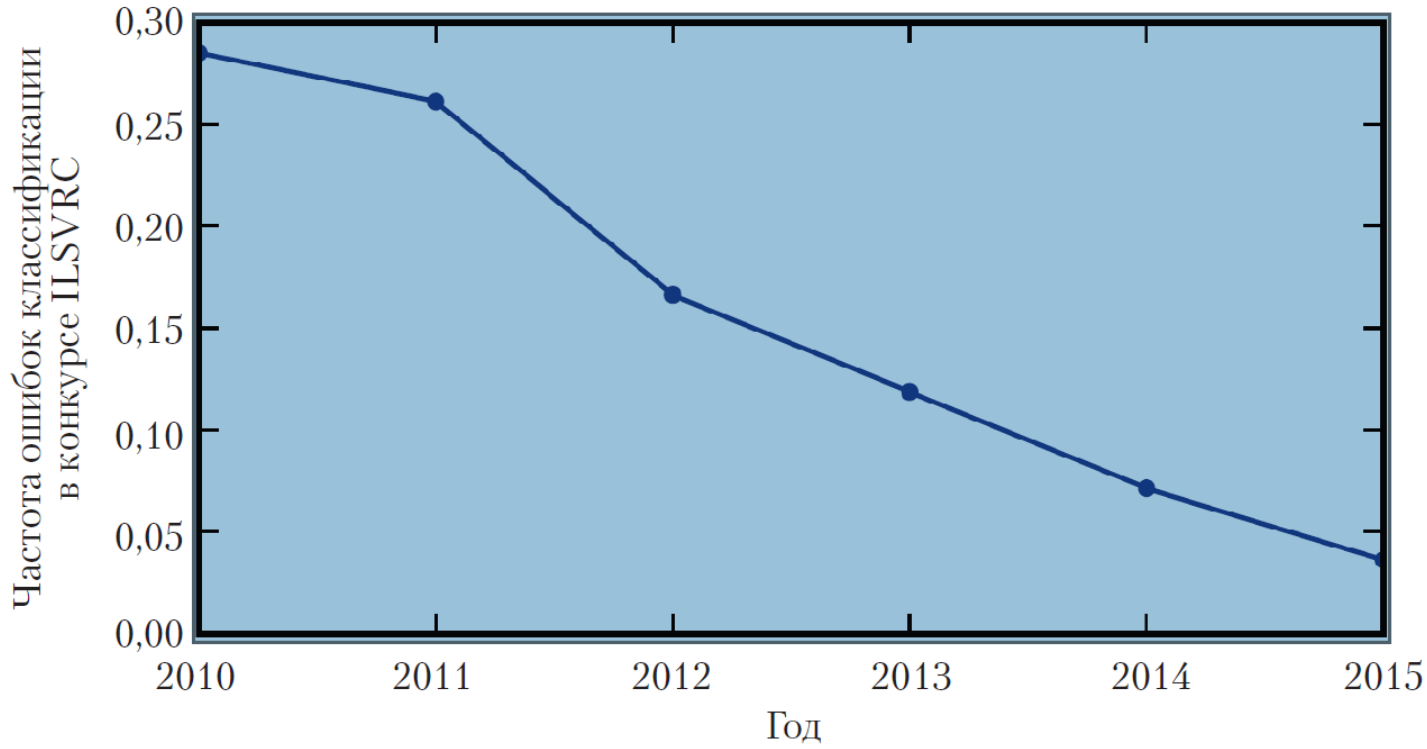
ПОВЫШЕНИЕ ТОЧНОСТИ И СЛОЖНОСТИ И РАСШИРЕНИЕ КРУГА ЗАДАЧ

Начиная с 1980-х годов точность распознавания и прогнозирования глубоких моделей постоянно росла. И вместе с тем все разнообразнее становились задачи, которые удавалось решать с их помощью.

Самые первые глубокие модели использовались для распознавания отдельных объектов в кадрированных изображениях совсем небольшого размера. С тех пор размер изображений, которые можно было обработать с помощью нейронной сети, постепенно увеличивался. Современные сети распознавания объектов обрабатывают фотографии с высоким разрешением и не требуют кадрирования фотографии по месту расположения объекта. Кроме того, ранние сети умели распознавать только два вида объектов (а в некоторых случаях присутствие или отсутствие объектов одного вида), тогда как типичная современная сеть распознает не менее 1000 категорий объектов. Самый крупный конкурс по распознаванию объектов – ImageNet Large Scale Visual Recognition Challenge (ILSVRC) – проводится каждый год. Переломным моментом, ознаменовавшим стремительный взлет глубокого обучения, стала победа с большим отрывом сверточной сети, которая участвовала впервые и сразу уменьшила частоту непопадания в первые пять (top-5 error rate) с 26,1 до 15,3% . Смысл этого показателя следующий: сверточная сеть порождала для каждого изображения ранжированный список возможных категорий, и правильная категория отсутствовала среди первых пяти элементов этого списка только в 15,3% тестовых примеров. С тех пор подобные конкурсы неизменно выигрывали сверточные сети, и на данный момент прогресс глубокого обучения позволил довести частоту непопадания в первые пять до 3,6%

Лекция 1.6 - Задачи искусственного интеллекта, требующие глубокого обучения

ПОВЫШЕНИЕ ТОЧНОСТИ И СЛОЖНОСТИ И РАСШИРЕНИЕ КРУГА ЗАДАЧ



Уменьшение частоты ошибок со временем. С тех пор как глубокие сети достигли масштаба, необходимого для участия в конкурсе ImageNet Large Scale Visual Recognition Challenge, они неизменно выигрывают его, с каждым разом демонстрируя все меньшую и меньшую частоту ошибок.

Лекция 1.6 - Задачи искусственного интеллекта, требующие глубокого обучения

ПОВЫШЕНИЕ ТОЧНОСТИ И СЛОЖНОСТИ И РАСШИРЕНИЕ КРУГА ЗАДАЧ

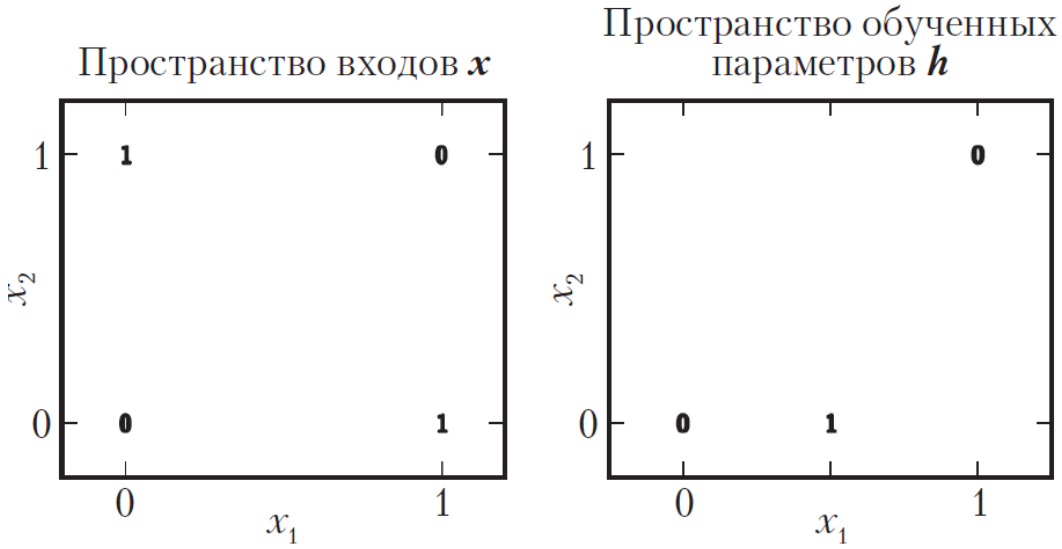
Глубокое обучение также оказало огромное влияние на распознавание речи. После прогресса, достигнутого на протяжении 1990-х годов, в качестве распознавания речи наступил застой. Применение глубокого обучения привело к резкому уменьшению частоты ошибок, иногда аж наполовину. Одновременно с повышением размера и точности глубоких сетей росла и сложность решаемых с их помощью задач. Нейронные сети можно научить распознаванию целых последовательностей символов в изображении, а не только идентификации одиночного объекта. Ранее считалось, что для такого обучения необходимо помечать отдельные элементы последовательности.

Рекуррентные нейронные сети, в частности модель LSTM, теперь применяются для моделирования связей последовательностей с другими последовательностями, а не только с фиксированными входами. Резюмируя, можно сказать, что глубокое обучение – это развивавшийся в течение нескольких десятилетий подход к машинному обучению, основанный главным образом на наших знаниях о человеческом мозге, на статистике и прикладной математике. В последние годы глубокое обучение переживает стремительный рост популярности и полезности благодаря в первую очередь появлению более мощных компьютеров, больших наборов данных и методов обучения более глубоких сетей. Будущее сулит нам новые проблемы и возможности дальнейшего совершенствования глубокого обучения с выходом на новые рубежи.

РАЗДЕЛ 2
СОВРЕМЕННЫЕ МОДЕЛИ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

Лекция 2.1 - Глубокие сети прямого распространения

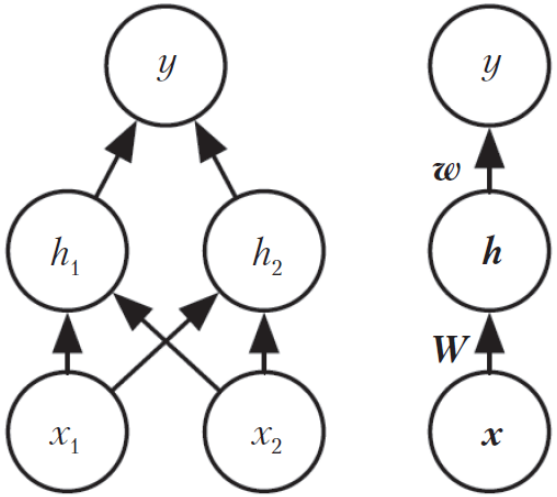
XOR



Решение задачи о функции XOR путем обучения представления. Жирными цифрами обозначены значения, которые обученная функция должна вывести в каждой точке. (Слева) Линейная модель, применяемая непосредственно к входным данным, неспособна реализовать функцию XOR. При $x_1 = 0$ выход модели должен возрастать с ростом x_2. При $x_1 = 1$ выход модели должен убывать с ростом x_2. В линейной модели должен быть фиксированный коэффициент между w_2 и x_2. Поэтому линейная модель не может использовать значение x_1 для изменения коэффициента при x_2 и, стало быть, не в состоянии решить задачу. (Справа) В преобразованном пространстве признаков, выделяемых нейронной сетью, линейная модель может решить задачу. В нашем случае обе точки, которые должны выводить 1, схлопнуты в одну точку пространства признаков. Иными словами, нелинейное преобразование отображает точки $x = [1, 0]^\top$ и $x = [0, 1]^\top$ в одну точку пространства признаков $h = [1, 0]^\top$. Теперь линейная модель может описать функцию, возрастающую относительно h_1 и убывающую относительно h_2. В этом примере причиной для обучения в пространстве признаков было всего лишь желание увеличить емкость модели, так чтобы она могла аппроксимировать обучающий набор. В реальных приложениях обученное таким образом представление способствует лучшей обобщаемости модели

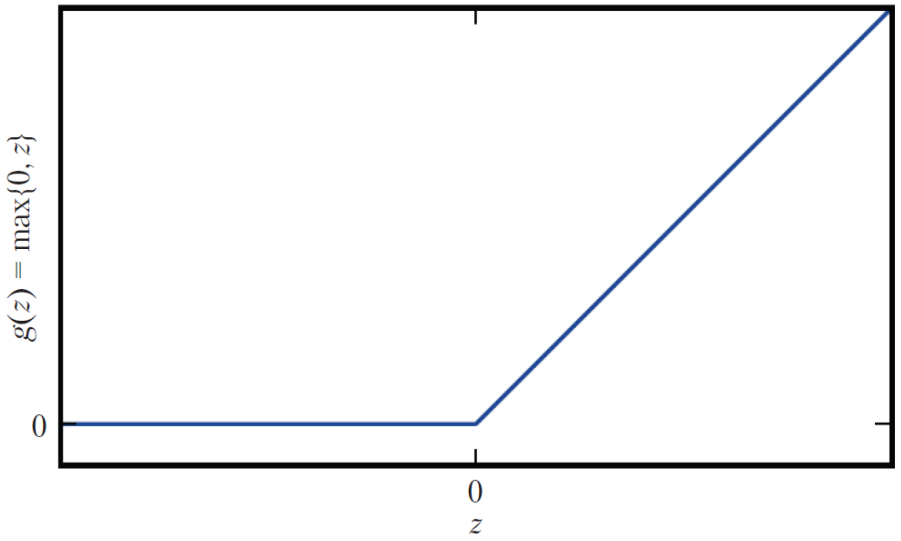
Лекция 2.1 - Глубокие сети прямого распространения

XOR



Пример сети обратного распространения, нарисованной двумя способами. Это та сеть, которой мы воспользовались для решения задачи о функции XOR. В сети имеется скрытый слой с двумя блоками. (Слева) Здесь каждый блок представлен вершиной графа. Этот способ явный и однозначный, но когда сеть побольше, чем в этом примере, рисунок занимает слишком много места. (Справа) В этом случае вершина графа соответствует целому вектору, представляющему все активации в слое. Такой способ гораздо компактнее. Иногда ребра графа аннотируются именами параметров, описывающих связь между двумя слоями. Здесь мы указали, что матрица W описывает отображение x в h, а вектор w – отображение h в y. Как правило, свободные члены для каждого слоя в таких метках опускаются.

Ректифицированная линейная функция активации. Эта функция по умолчанию рекомендуется для большинства нейронных сетей прямого распространения. Применение ее к выходу линейного преобразования дает нелинейное преобразование. Функция, впрочем, очень близка к линейной – она является кусочно-линейной с двумя линейными участками. Поскольку блоки линейной ректификации почти линейны, сохраняются многие свойства, благодаря которым линейные модели легко поддаются оптимизации градиентными методами. Сохраняются также свойства, обеспечивающие хорошую обобщаемость линейных моделей.



Лекция 2.1 - Глубокие сети прямого распространения

ОБУЧЕНИЕ ГРАДИЕНТНЫМИ МЕТОДАМИ

Основное различие между линейными моделями, встречавшимися нам до сих пор, и нейронными сетями состоит в том, что из-за нелинейности нейронной сети большинство интересных функций потерь оказывается невыпуклыми. Это означает, что нейронные сети обычно обучаются с помощью итеративных градиентных оптимизаторов, которые просто находят очень малое значение функции стоимости, а не с помощью методов решения линейных уравнений, применяемых при обучении моделей регрессии, или алгоритмов выпуклой оптимизации, гарантированно сходящихся к глобальному оптимуму, которые используются при обучении логистической регрессии или модели опорных векторов. Важный аспект проектирования глубокой нейронной сети – выбор функции стоимости. По счастью, функции стоимости для нейронных сетей мало отличаются от применяемых в других параметрических моделях, в т. ч. линейных.

ОБУЧЕНИЕ УСЛОВНЫХ РАСПРЕДЕЛЕНИЙ С ПОМОЩЬЮ МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ

Преимущество такого подхода – вывода функции стоимости из оценки максимального правдоподобия – в том, что отпадает необходимость проектировать функцию стоимости заново для каждой модели. Задание модельного распределения $p(y \mid x)$ автоматически определяет функцию стоимости $\log p(y \mid x)$.

ОБУЧЕНИЕ УСЛОВНЫХ СТАТИСТИК

Вместо обучения полного распределения вероятности $p(y \mid x; \theta)$ мы часто хотим обучить только одну условную статистику y при заданном x. Для решения задачи оптимизации на множестве функций имеется специальный математический аппарат – вариационное Исчисление.

Лекция 2.1 - Глубокие сети прямого распространения

ВЫХОДНЫЕ БЛОКИ

Выбор функции стоимости тесно связан с выбором выходного блока. Как правило, мы просто используем перекрестную энтропию между распределением данных и модельным распределением. Выбор представления выхода определяет форму функции перекрестной энтропии. Любой блок нейронной сети, который можно использовать в качестве выходного, годится и на роль скрытого.

ЛИНЕЙНЫЕ БЛОКИ ДЛЯ НОРМАЛЬНЫХ ВЫХОДНЫХ РАСПРЕДЕЛЕНИЙ

Простой выходной блок основан на аффинном преобразовании без нелинейностей. Часто такие блоки называются просто линейными.

СИГМОИДНЫЕ БЛОКИ И ВЫХОДНОЕ РАСПРЕДЕЛЕНИЕ БЕРНУЛЛИ

Во многих задачах требуется предсказать значение бинарной величины y. В таком виде можно представить задачу классификации с двумя классами. Подход на основе максимального правдоподобия заключается в определении распределения Бернулли величины y при условии x.

БЛОКИ SOFTMAX И КАТЕГОРИАЛЬНОЕ ВЫХОДНОЕ РАСПРЕДЕЛЕНИЕ

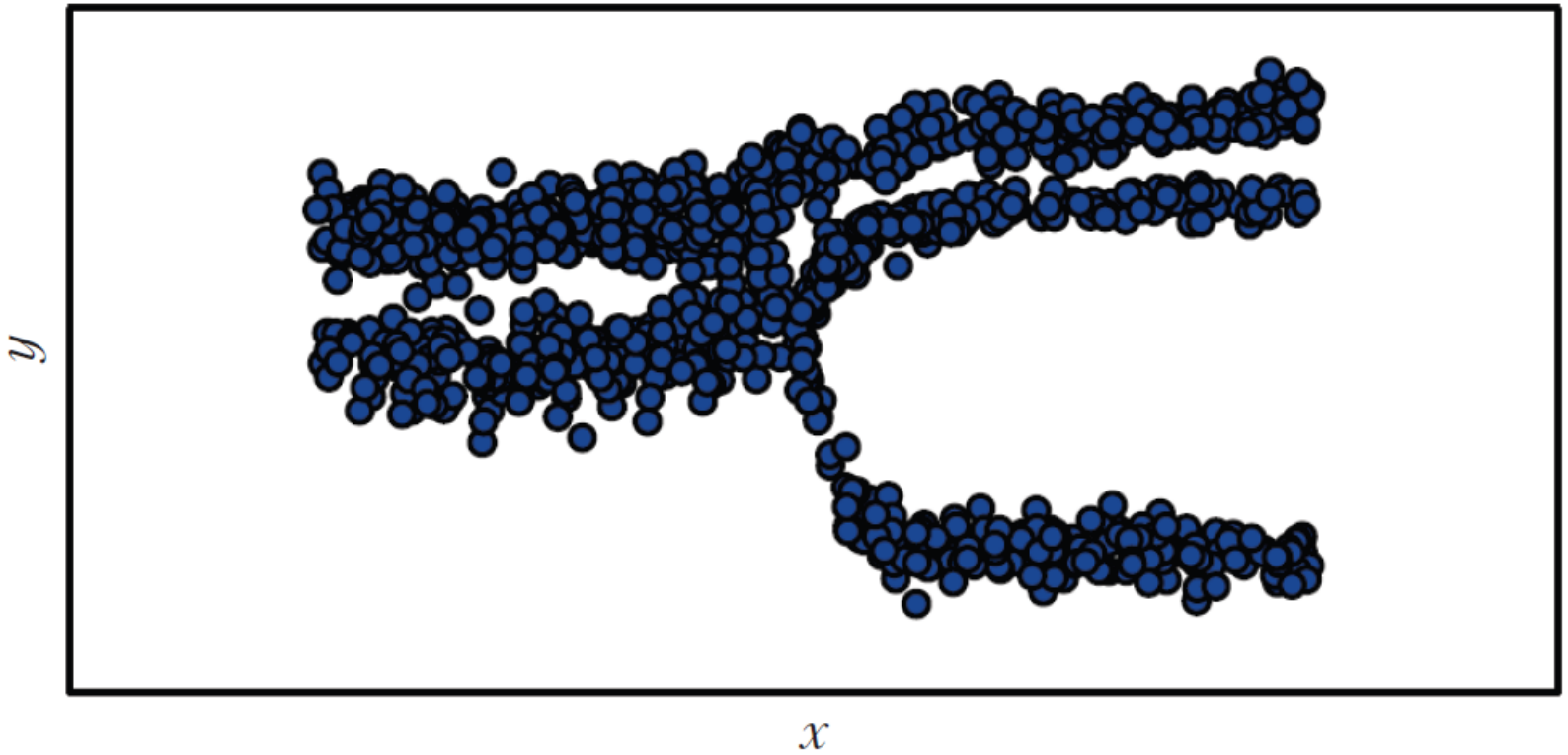
Если требуется представить распределение вероятности дискретной величины, принимающей n значений, то можно воспользоваться функцией softmax. Ее можно рассматривать как обобщение сигмоиды, которая использовалась для представления распределения бинарной величины.

ДРУГИЕ ТИПЫ ВЫХОДНЫХ БЛОКОВ

Описанные выше выходные блоки – линейные, сигмоидные и softmax – применяются наиболее часто. Но нейронные сети обобщаются практически на любой желаемый выходной слой. Принцип максимального правдоподобия подсказывает, как спроектировать хорошую функцию стоимости.

Лекция 2.1 - Глубокие сети прямого распространения

ВЫХОДНЫЕ БЛОКИ



Примеры получены от нейронной сети с выходным слоем в виде смеси распределений. Вход x выбирается из равномерного распределения, а выход y – из $p_{model}(y \mid x)$. Нейронная сеть способна обучить нелинейные отображения входа на параметры выходного распределения. В состав этих параметров входят вероятности, управляющие тем, какая из трех компонент смеси порождает выход, а также параметры отдельных компонент. Каждая компонента смеси – нормальное распределение с предсказанными средним и дисперсией. Все эти аспекты выходного распределения могут изменяться в зависимости от x, причем нелинейно

Лекция 2.1 - Глубокие сети прямого распространения

СКРЫТЫЕ БЛОКИ

Проектирование скрытых блоков – чрезвычайно активная область исследований, в которой не так уж много руководящих теоретических принципов.

БЛОКИ ЛИНЕЙНОЙ РЕКТИФИКАЦИИ И ИХ ОБОБЩЕНИЯ

В блоке линейной ректификации используется функция активации $g(z) = \max\{0, z\}$. Эти блоки легко оптимизировать, потому что они очень похожи на линейные. Разница только в том, что блок линейной ректификации в половине своей области определения выводит 0. Блоки линейной ректификации обычно применяются после аффинного преобразования.

ЛОГИСТИЧЕСКАЯ СИГМОИДА И ГИПЕРБОЛИЧЕСКИЙ ТАНГЕНС

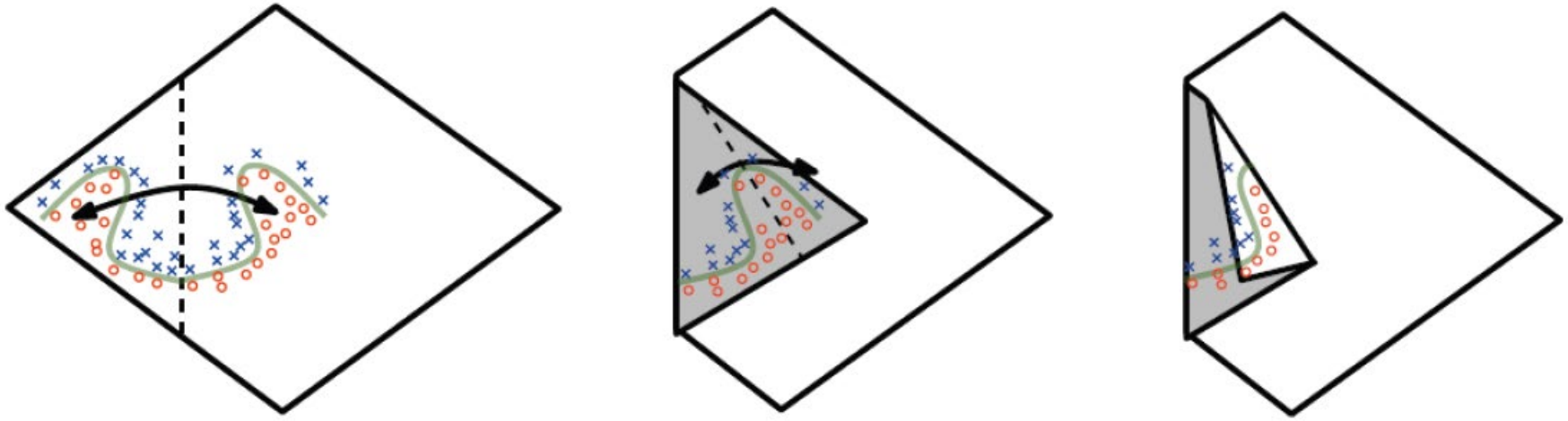
Блоки линейной ректификации стали использоваться сравнительно недавно, а раньше в большинстве нейронных сетей в роли функции активации применялась логистическая сигмоида

ДРУГИЕ СКРЫТЫЕ БЛОКИ

Существует много других типов скрытых блоков, но используются они реже. Вообще говоря, многие дифференцируемые функции показывают отличные результаты. Есть целый ряд неопубликованных функций активации, которые ведут себя ничуть не хуже популярных.

Лекция 2.1 - Глубокие сети прямого распространения

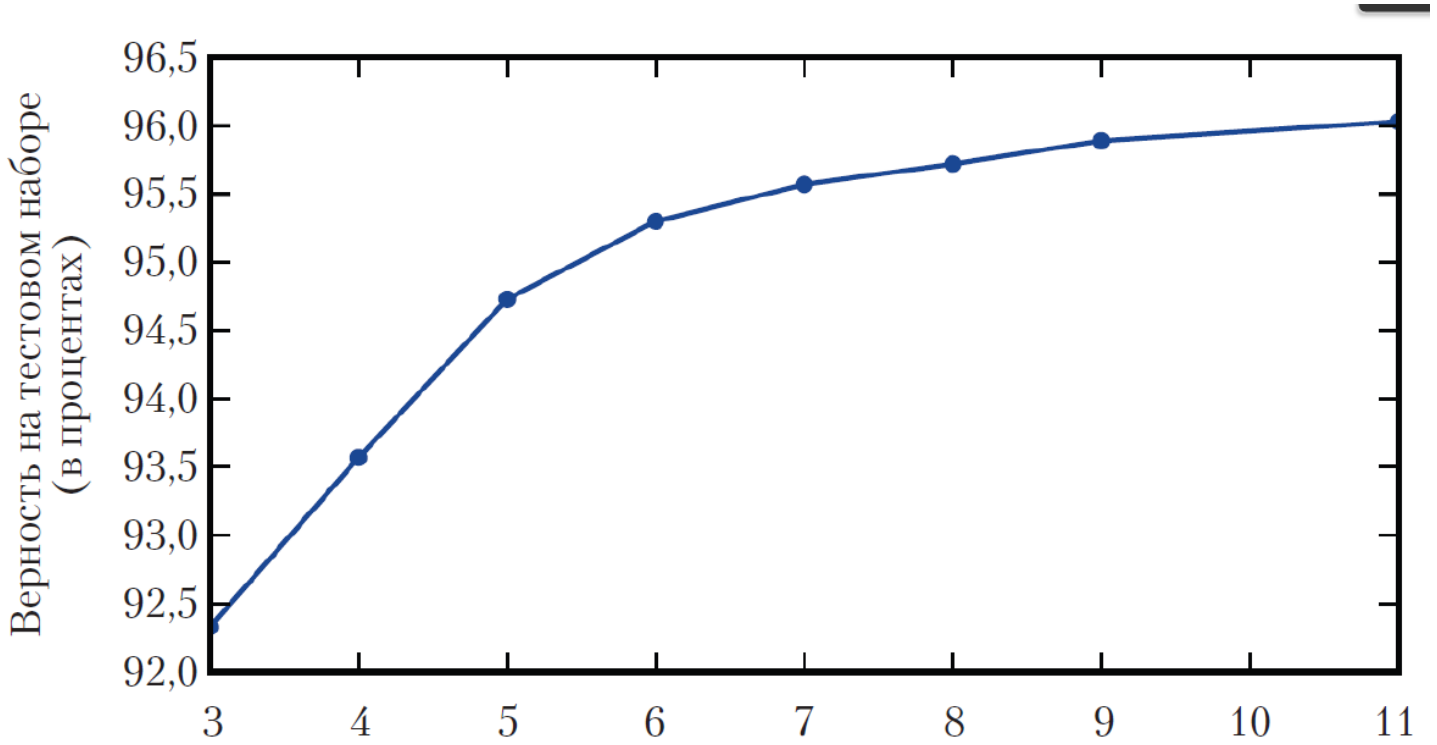
СВОЙСТВА УНИВЕРСАЛЬНОЙ АППРОКСИМАЦИИ И ГЛУБИНА



Интуитивное геометрическое объяснение экспоненциального характера глубоких сетей ректификаторов. (Слева) Блок абсолютной ректификации порождает одинаковые выходы для любой пары зеркально симметричных входов. Ось симметрии задается гиперплоскостью, определяемой весами и смещением блока. Функция, вычисляемая этим блоком (зеленая решающая поверхность), будет зеркальным отражением более простого паттерна относительно этой оси симметрии. (В центре) Функцию можно получить путем перегибания пространства по этой оси симметрии. (Справа) На первый повторяющийся паттерн можно наложить еще один (с помощью блока следующего слоя) и получить тем самым еще одну симметрию (четырехкратное повторение при двух скрытых слоях).

Лекция 2.1 - Глубокие сети прямого распространения

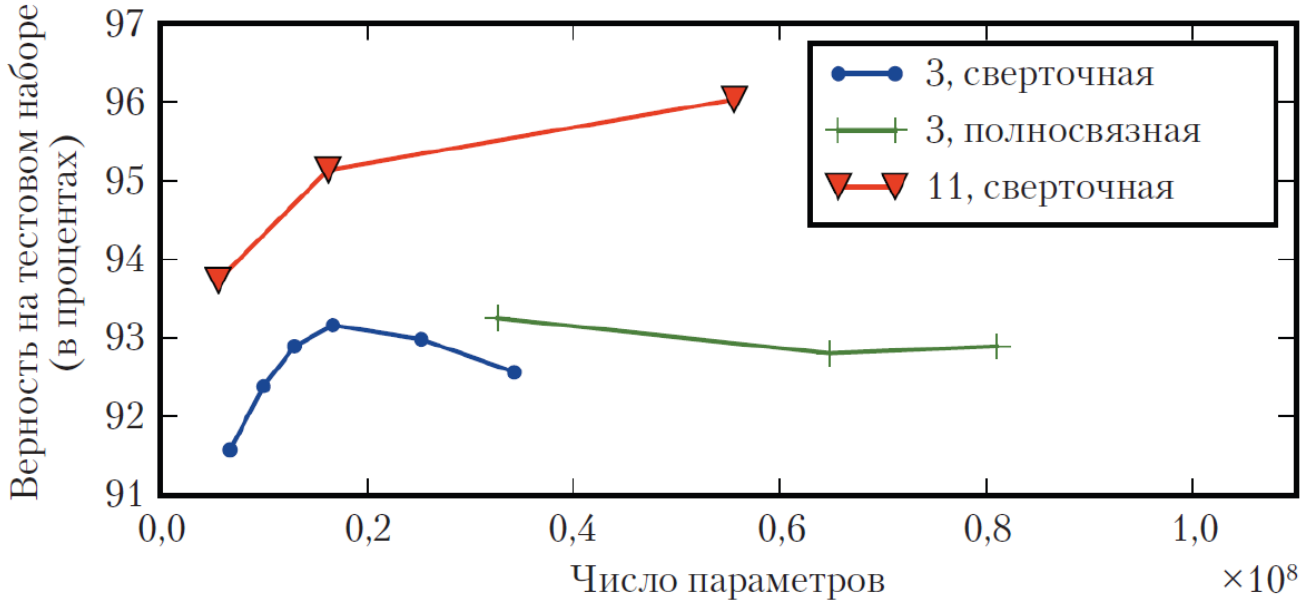
СВОЙСТВА УНИВЕРСАЛЬНОЙ АППРОКСИМАЦИИ И ГЛУБИНА



Влияние глубины. Эмпирические результаты показывают, что более глубокие сети лучше обобщаются в задаче распознавания нескольких цифр на фотографиях адресов. Данные взяты из работы Goodfellow et al. (2014d). Верность на тестовом наборе монотонно растет при увеличении глубины.

Лекция 2.1 - Глубокие сети прямого распространения

ДРУГИЕ АРХИТЕКТУРНЫЕ ПОДХОДЫ



Влияние числа параметров. Более глубокие модели обычно работают лучше. Но это не просто потому, что модель больше. Увеличение числа параметров в слоях сверточной сети, не сопровождаемое увеличением ее глубины, далеко не так эффективно с точки зрения обобщения на тестовый набор. В надписях на рисунке указана глубина сети, соответствующей каждой кривой, и что именно отражает кривая: изменение размера сверточных или полносвязных слоев. Мы видим, что мелкие модели в этом контексте оказываются переобучены, когда число параметров составляет примерно 20 миллионов, а качество глубоких продолжает улучшаться вплоть до числа параметров порядка 60 миллионов. Это позволяет предположить, что глубокая модель выражает полезную гипотезу о пространстве обучаемых ей функций, а именно она выражает веру в то, функция должна быть образована композицией многих более простых функций. Результатом может быть либо обучение представления, составленного из более простых представлений (например, углы, определяемые в терминах границ), либо обучение программы, состоящей из последовательных зависимых друг от друга шагов (например, сначала найти множество объектов, затем сегментировать их, отделив друг от друга, и потом распознать их)

Лекция 2.2 - Регуляризация в глубоком обучении

Центральная проблема машинного обучения – как создать алгоритм, который будет хорошо работать не только на обучающих, но и на новых данных. Многие используемые стратегии специально предназначены для уменьшения ошибки тестирования, быть может, за счет увеличения ошибки обучения. Эти стратегии известны под общим названием «***регуляризация***».

Смысл регуляризация оценки – в увеличении смещения в обмен на уменьшение дисперсии. Эффективным считается регуляризатор, который находит выгодный компромисс, т. е. значительно уменьшает дисперсию, не слишком увеличивая смещение.

ШТРАФЫ ПО НОРМЕ ПАРАМЕТРОВ

Регуляризованная целевая функция: $$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

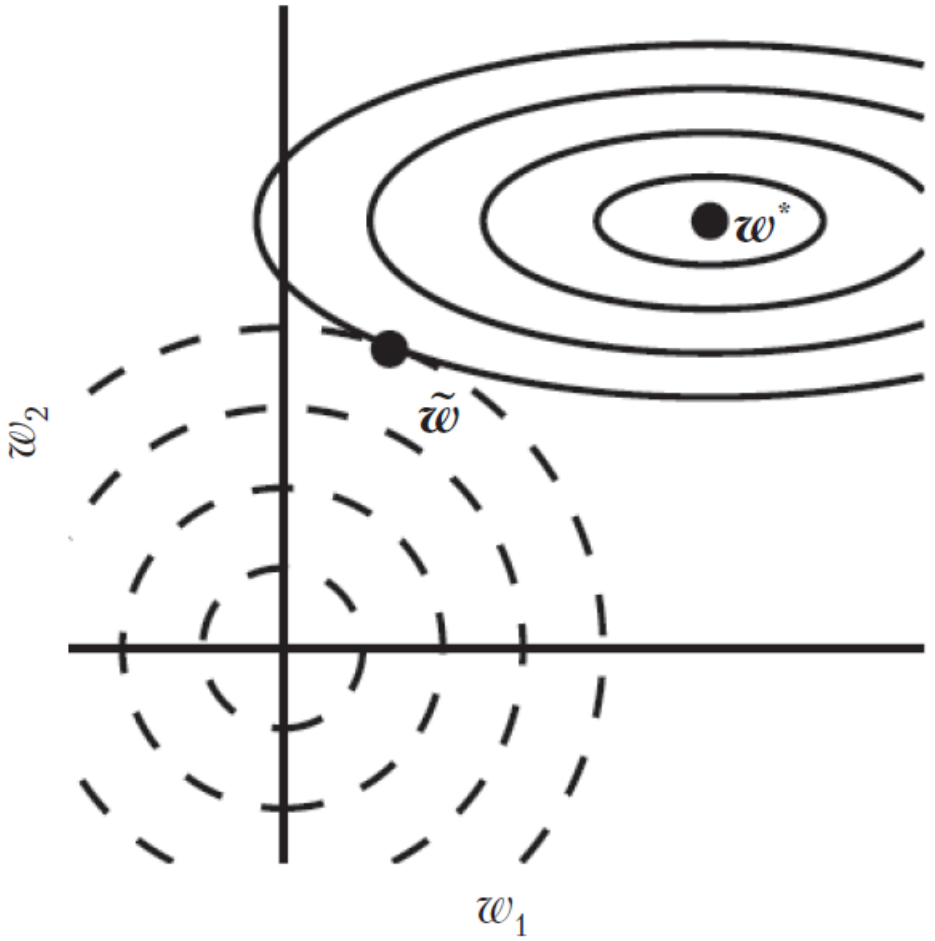
Минимизируя регуляризованную целевую функцию J, наш алгоритм обучения одновременно уменьшает исходную целевую функцию J на обучающих данных и некоторую меру величины параметров θ (или какого-то подмножества параметров).

В контексте нейронных сетей иногда желательно использовать штрафы с разными коэффициентами α в каждом слое. Поскольку подбор правильных значений нескольких гиперпараметров иногда обходится дорого, все равно разумно использовать одно и то же снижение весов во всех слоях, просто чтобы уменьшить размер пространства поиска.

Лекция 2.2 - Регуляризация в глубоком обучении

РЕГУЛЯРИЗАЦИЯ ПАРАМЕТРОВ ПО НОРМЕ L^2

Иллюстрация влияния регуляризации по норме L2 (снижения весов) на значение оптимального вектора w. Сплошными эллипсами представлены линии равных значений нерегуляризированной целевой функции, а пунктирными – линии равных значений L^2-регуляризатора. В точке w~ эти конкурирующие цели достигают равновесия. По первому измерению собственное значение гессиана J мало. Целевая функция слабо растет при удалении от w* по горизонтали. Поскольку целевая функция не выказывает сильного предпочтения этому направлению, регуляризатор дает для него значительный эффект. Регуляризатор подтягивает w_1 ближе к нулю. По второму направлению целевая функция очень чувствительна к удалению от w*. Соответствующее собственное значение велико, что указывает на сильную кривизну. Поэтому снижение весов влияет на положение w_2 сравнительно слабо



Лекция 2.2 - Регуляризация в глубоком обучении

L^1-РЕГУЛЯРИЗАЦИЯ

Формально L^1-регуляризация параметров модели w определяется по формуле:

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

Свойство разреженности, присущее L^1-регуляризации, активно эксплуатировалось как механизм отбора признаков, идея которого состоит в том, чтобы упростить задачу машинного обучения за счет выбора некоторого подмножества располагаемых признаков. В частности, хорошо известная модель LASSO объединяет L^1-штраф с линейной моделью и среднеквадратической функцией стоимости. Благодаря L^1-штрафу некоторые веса обращаются в 0, и соответствующие им признаки отбрасываются.

ШТРАФ ПО НОРМЕ КАК ОПТИМИЗАЦИЯ С ОГРАНИЧЕНИЯМИ

Причина использовать явные ограничения и обратное проецирование, а не косвенные ограничения в виде штрафов, состоит в том, что из-за штрафа процедура невыпуклой оптимизации может застрять в локальном минимуме, соответствующем малому θ. При обучении нейронных сетей это обычно проявляется в виде сети, обучившейся с несколькими «мертвыми блоками». Так называются блоки, которые мало что вносят в поведение обученной сетью функции, потому что веса всех входных или выходных сигналов очень малы. При обучении со штрафом на норму весов такие конфигурации могут оказаться локально оптимальными, даже если возможно значительно уменьшить J, сделав веса больше. Явные ограничения, реализованные с помощью обратного проецирования, в таких случаях могут работать гораздо лучше, потому что не поощряют приближения весов к началу координат. Такие явные ограничения вступают в силу, только когда веса становятся большими и грозят выйти за пределы области ограничений.

Лекция 2.2 – Регуляризация в глубоком обучении

РЕГУЛЯРИЗАЦИЯ И НЕДООПРЕДЕЛЕННЫЕ ЗАДАЧИ

В некоторых случаях без регуляризации просто невозможно корректно поставить задачу машинного обучения. Многие линейные модели в машинном обучении, в т. ч. линейная регрессия и PCA, включают обращение матрицы $X^\top X$. Это невозможно, если $X^\top X$ сингулярна. Матрица может оказаться сингулярной, если порождающее распределение действительно не имеет дисперсии в некотором направлении или если в некотором направлении не наблюдается дисперсии, потому что число примеров (строк X) меньше числа входных признаков (столбцов X). В таком случае во многих вариантах регуляризации прибегают к обращению матрицы $X^\top X + \alpha I$. Гарантируется, что такая регуляризированная матрица обратима.

РОБАСТНОСТЬ ОТНОСИТЕЛЬНО ШУМА

Для некоторых моделей добавление шума с очень малой дисперсией к входу модели эквивалентно штрафованию по норме весов. В общем случае важно помнить, что привнесение шума может дать куда больший эффект, чем простое уменьшение параметров, особенно если шум примешивается к скрытым блокам.

ПРИВНЕСЕНИЕ ШУМА В ВЫХОДНЫЕ МЕТКИ

В большинстве наборов данных выходные метки y содержат сколько-то ошибок. Максимизация $\log p(y \mid x)$, когда y ошибочно, чревата неприятными последствиями. Предотвратить это можно, в частности, путем явного моделирования шума в метках.

Лекция 2.3 - Оптимизация в обучении глубоких моделей

Алгоритмы глубокого обучения включают оптимизацию в самых разных контекстах. Например, для выполнения вывода в таких моделях, как метод главных компонент, необходимо решать задачу оптимизации. Мы часто применяем аналитическую оптимизацию для доказательства правильности или проектирования алгоритмов. Из всех многочисленных задач оптимизации, решаемых в глубоком обучении, самые трудные возникают при обучении нейронной сети.

ЧЕМ ОБУЧЕНИЕ ОТЛИЧАЕТСЯ ОТ ЧИСТОЙ ОПТИМИЗАЦИИ

Алгоритмы оптимизации, используемые для обучения глубоких моделей, отличаются от традиционных алгоритмов оптимизации в нескольких отношениях. Машинное обучение обычно работает не напрямую. В большинстве ситуаций нас интересует некоторая мера качества P, которая определена относительно тестового набора и может оказаться вычислительно неприступной. Поэтому мы оптимизируем P косвенно. Мы уменьшаем другую функцию стоимости J(θ) в надежде, что при этом улучшится и P. Это резко отличается от чистой оптимизации, где минимизация J и есть конечная цель. Кроме того, алгоритмы оптимизации для обучения глубоких моделей обычно включают специализации для конкретной структуры целевых функций.

МИНИМИЗАЦИЯ ЭМПИРИЧЕСКОГО РИСКА

$$\mathbb{E}_{\boldsymbol{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x};\boldsymbol{\theta}),y)]=\frac{1}{m}\sum_{i=1}^{m}L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),y^{(i)})$$

Лекция 2.3 - Оптимизация в обучении глубоких моделей

СУРРОГАТНЫЕ ФУНКЦИИ ПОТЕРЬ И РАННЯЯ ОСТАНОВКА

Очень важное различие между оптимизацией вообще и применяемой в алгоритмах обучения заключается в том, что алгоритмы обучения обычно останавливаются не в локальном минимуме. Вместо этого алгоритм, как правило, минимизирует суррогатную функцию потерь, но останавливается, когда выполнено условие сходимости, основанное на идее ранней остановки. Типичное условие ранней остановки основано на истинной функции потерь, например вычислении бинарной функции потерь на контрольном наборе, и предназначено для того, чтобы остановить работу алгоритма, когда возникает угроза переобучения. Обучение зачастую заканчивается, когда производные суррогатной функции потерь все еще велики, и этим разительно отличается от чистой оптимизации, при которой считается, что алгоритм сошелся, если градиент стал очень малым.

ПАКЕТНЫЕ И МИНИ-ПАКЕТНЫЕ АЛГОРИТМЫ

Еще одно отличие алгоритмов машинного обучения от общих алгоритмов оптимизации состоит в том, что целевая функция обычно представлена в виде суммы по обучающим примерам. Алгоритмы оптимизации, в которых используется весь обучающий пакет, называются ***пакетными***, или ***детерминированными***, градиентными методами, поскольку обрабатывают сразу все примеры одним большим пакетом.

Алгоритмы оптимизации, в которых используется по одному примеру за раз, иногда называют ***стохастическими***, или ***онлайновыми***, методами. Термин «онлайновый» обычно резервируется для случая, когда примеры выбираются из непрерывного потока, а не из обучающего набора фиксированного размера, по которому можно совершать несколько проходов.

Лекция 2.3 - Оптимизация в обучении глубоких моделей

ПАКЕТНЫЕ И МИНИ-ПАКЕТНЫЕ АЛГОРИТМЫ

Большинство алгоритмов, используемых в глубоком обучении, находится где-то посередине – число примеров в них больше одного, но меньше размера обучающего набора. Традиционно они назывались мини-пакетными, или мини-пакетными стохастическими, методами, а сейчас – просто стохастическими.

На размер мини-пакета оказывают влияние следующие факторы:

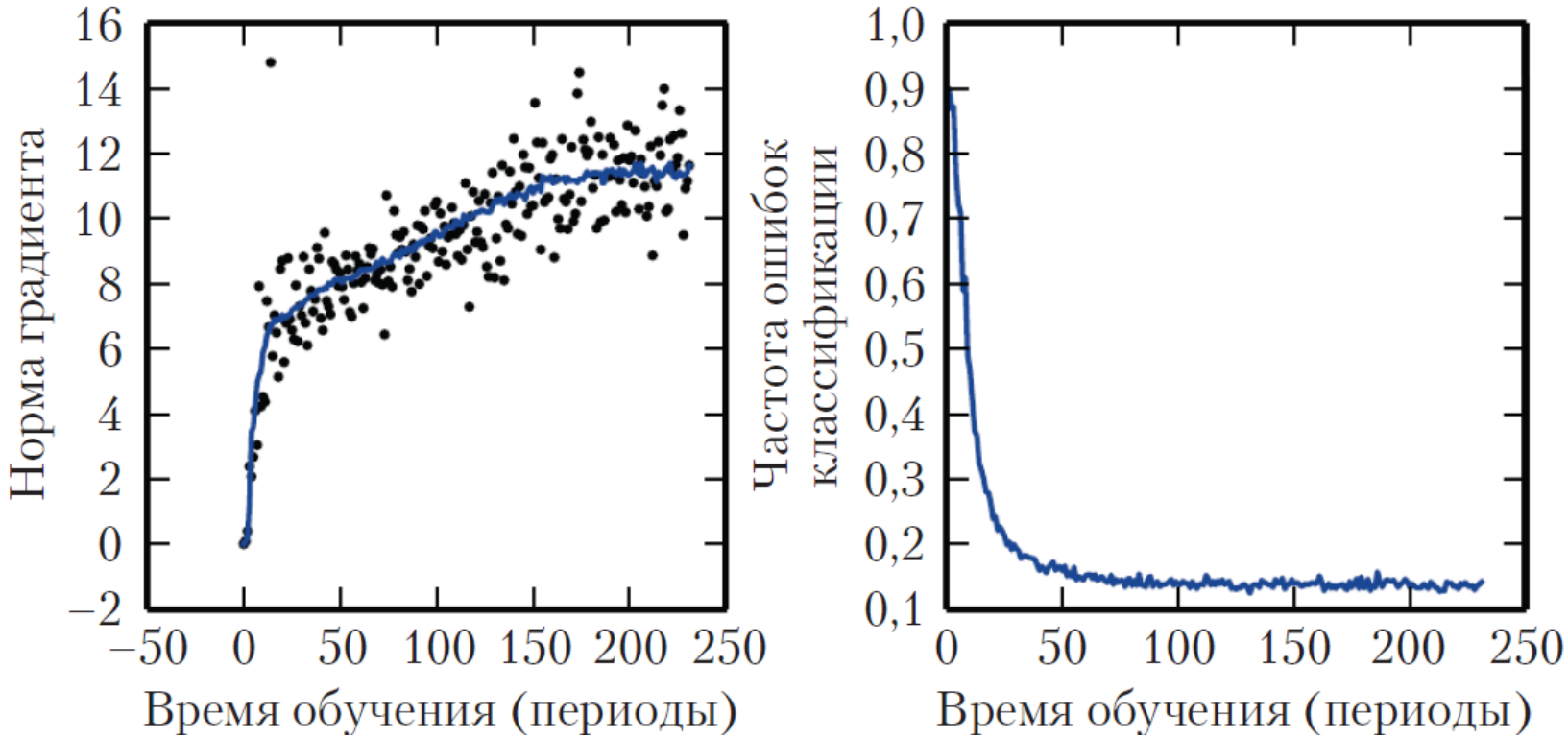
- Чем больше пакет, тем точнее оценка градиента, но зависимость хуже линейной;
- Если пакет очень мал, то не удается в полной мере задействовать преимущества многоядерной архитектуры. Поэтому существует некий абсолютный минимум размера пакета – такой, что обработка мини-пакетов меньшего размера не дает никакого выигрыша во времени;
- Если все примеры из пакета нужно обрабатывать параллельно (так обычно и бывает), то размер пакета лимитирован объемом памяти. Для многих аппаратных конфигураций размер пакета – ограничивающий фактор;
- Для некоторых видов оборудования оптимальное время выполнения достигается при определенных размерах массива. Так, для GPU наилучшие результаты получаются, когда размер пакета – степень 2. Типичный пакет имеет размер от 32 до 256, а для особо больших моделей иногда пробуют 16;
- Небольшие пакеты могут дать эффект регуляризации быть может, из-за шума, который они вносят в процесс обучения. Ошибка обобщения часто оказывается наилучшей для пакета размера 1. Но для обучения с таким маленьким размером пакета нужна небольшая скорость обучения для обеспечения устойчивости из-за высокой дисперсии оценки градиента. Общее время работы может оказаться очень большим из-за увеличения числа шагов – как из-за пониженной скорости обучения, так и потому, что для перебора всего обучающего набора требуется больше шагов.

Лекция 2.3 - Оптимизация в обучении глубоких моделей

ПРОБЛЕМЫ ОПТИМИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ

ПЛОХАЯ ОБУСЛОВЛЕННОСТЬ

Градиентный спуск часто не находит никакой критической точки. В этом примере норма градиента возрастает на протяжении всего процесса обучения сверточной нейронной сети для обнаружения объектов. (Слева) На диаграмме рассеяния показана зависимость вычисленной нормы градиента от времени. Для наглядности показана лишь одна норма на каждый период. Скользящее среднее всех норм градиента показано сплошной линией. Очевидно, что норма градиента со временем возрастает, а не убывает, как было бы, если бы процесс обучения сходился к критической точке. (Справа) Несмотря на возрастание градиента, процесс обучения можно считать успешным. Ошибка классификации на контрольном наборе убывает до низкого уровня



Лекция 2.3 - Оптимизация в обучении глубоких моделей

ПРОБЛЕМЫ ОПТИМИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ

ЛОКАЛЬНЫЕ МИНИМУМЫ

Нейронные сети и вообще любые модели с несколькими эквивалентно параметризованными латентными переменными имеют несколько локальных минимумов из-за проблемы идентифицируемости модели. Говорят, что модель идентифицируемая, если существует достаточно большой обучающий набор, который может исключить все конфигурации параметров модели, кроме одной. Модели с латентными переменными часто не являются идентифицируемыми, потому что мы можем получить эквивалентные модели, меняя латентные переменные местами.

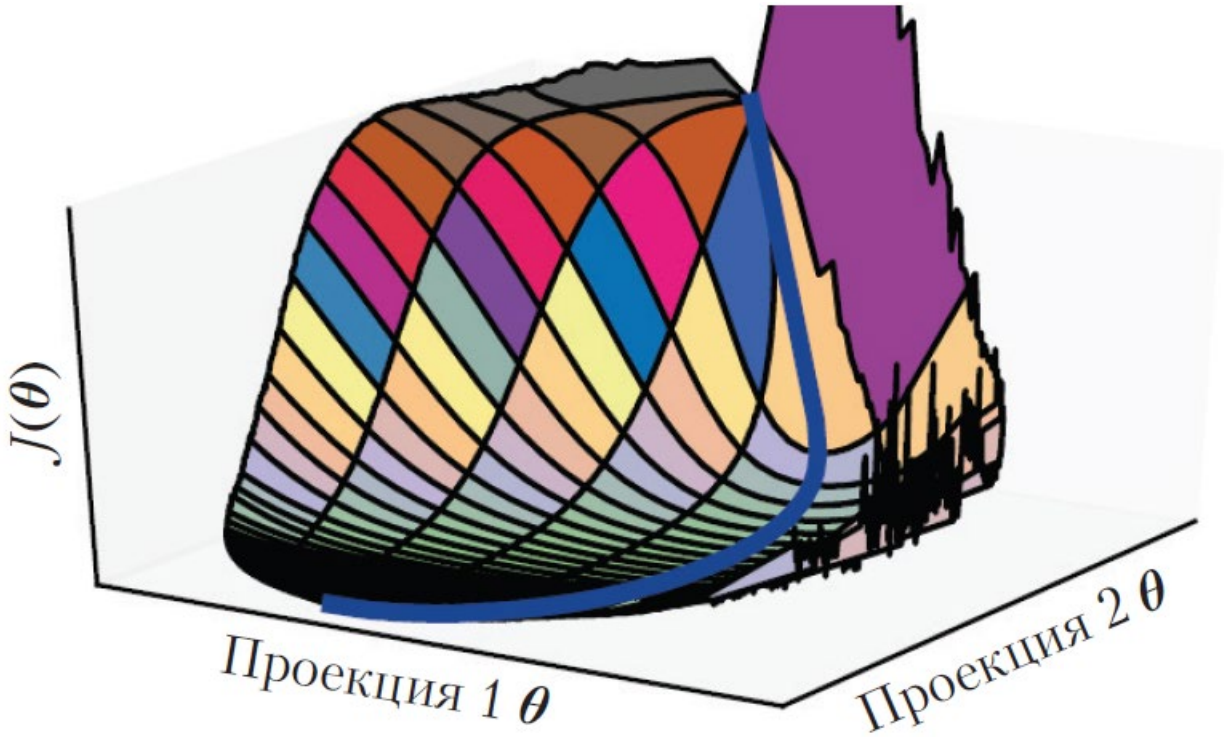
ПЛАТО, СЕДЛОВЫЕ ТОЧКИ И ДРУГИЕ ПЛОСКИЕ УЧАСТКИ

Для многих невыпуклых функций в многомерных пространствах локальные минимумы (и максимумы) встречаются гораздо реже других точек с нулевым градиентом: седловых точек. В одних точках в окрестности седловой стоимость выше, чем в седловой точке, в других – ниже. В седловой точке матрица Гессе имеет как положительные, так и отрицательные собственные значения. В точках, лежащих вдоль собственных векторов с положительными собственными значениями, стоимость выше, чем в седловой точке, а в точках, лежащих вдоль собственных векторов с отрицательными собственными значениями, – ниже. Можно считать, что седловая точка является локальным минимумом в одном сечении графика функции стоимости и локальным максимумом – в другом.

Лекция 2.3 – Оптимизация в обучении глубоких моделей

ПРОБЛЕМЫ ОПТИМИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ

Визуализация функции стоимости нейронной сети. Похожие визуализации характерны для нейронных сетей прямого распространения, а также сверточных и рекуррентных, применяемых в реальных задачах распознавания объектов и обработки естественных языков. Как ни странно
на этих визуализациях обычно не встретишь много бросающихся в глаза препятствий. До триумфа алгоритма стохастического градиентного спуска в применении к обучению очень больших моделей, датируемого примерно 2012 годом, считалось, что поверхности функций стоимости нейронных сетей обладают куда более невыпуклой структурой, чем видно на этих проекциях. Основное присутствующее здесь препятствие – седловая точка высокой стоимости вблизи начальных значений параметров, но, как показывает синяя линия, траектория обучения СГС быстро покидает эту седловую точку. Основное время затрачено на пересечение сравнительно плоской долины функции стоимости, наверное, вследствие высокого шума при вычислении градиента, плохой обусловленности гессиана в этой области или просто из-за необходимости обойти высокую «гору», видную на рисунке, по огибающей дуге.

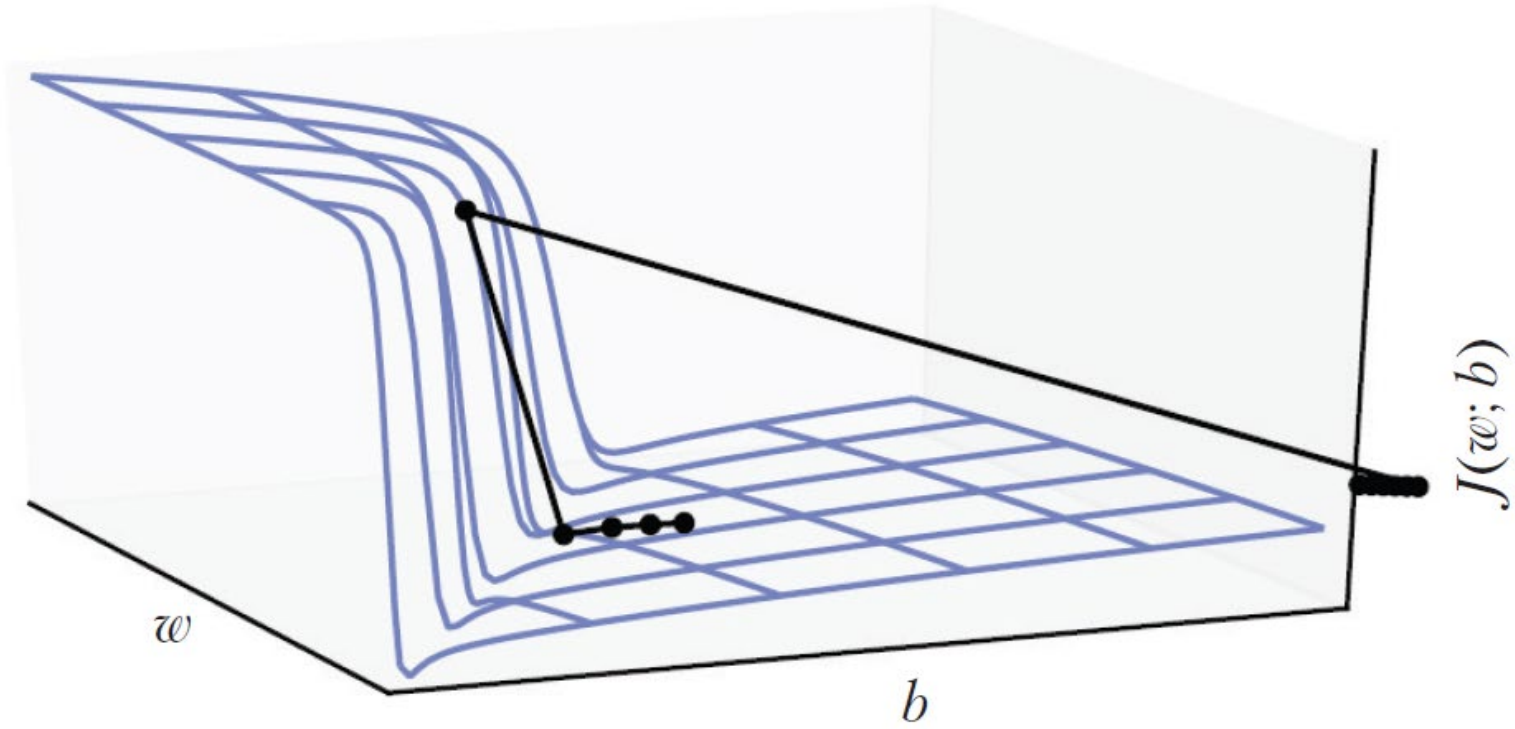


Лекция 2.3 - Оптимизация в обучении глубоких моделей

ПРОБЛЕМЫ ОПТИМИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ

ДОЛГОСРОЧНЫЕ ЗАВИСИМОСТИ

Целевая функция сильно нелинейной глу
нейронной сети или рекуррентной сети
характеризуется резкими нелинейностями
пространстве параметров, возникающими
перемножения нескольких параметров. В местах
нелинейностей значения производных очень в
Когда параметры приближаются к подобному утес
обновления в методе градиентного спуска
сдвинуть параметры очень далеко, при этом
потеряться все, чего удалось достичь в
предшествующей оптимизации.

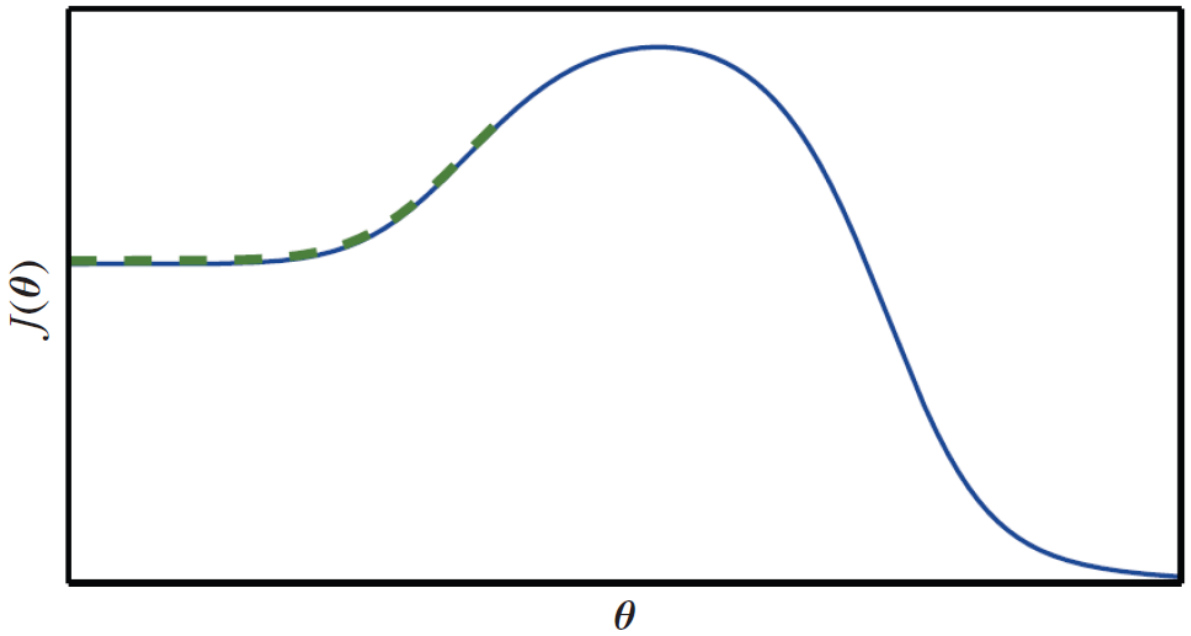


Лекция 2.3 - Оптимизация в обучении глубоких моделей

ПРОБЛЕМЫ ОПТИМИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ

ПЛОХОЕ СООТВЕТСТВИЕ МЕЖДУ ЛОКАЛЬНОЙ И ГЛОБАЛЬНОЙ СТРУКТУРАМИ

Оптимизация, основанная на локальном спуске, может потерпеть неудачу, если локальное направление не ведет к глобальному решению. Здесь показано, как такое может произойти даже в отсутствие локальных минимумов или седловых точек. Функция стоимости в этом примере только асимптотически приближается к низким значениям, но не имеет минимумов. Проблема вызвана тем, что начальные значения выбраны не по ту сторону «горы», и алгоритм не может перебраться через нее. В многомерных пространствах алгоритмы обучения обычно способны обогнуть такие горы, но траектория может оказаться длинной, и обучение займет слишком много времени



Лекция 2.4 - Сверточные сети

Сверточные сети – это просто нейронные сети, в которых вместо общей операции умножения на матрицу, по крайней мере в одном слое, используется свертка.

ОПЕРАЦИЯ СВЕРТКИ

В самом общем виде ***свертка*** – это операция над двумя функциями вещественного аргумента. В терминологии сверточных сетей первый аргумент (в нашем примере функция x) называется ***входом***, а второй (функция w) – ***ядром***. Выход иногда называют ***картой признаков***.

Дискретная свертка:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

В приложениях машинного обучения входом обычно является многомерный массив данных, а ядром – многомерный массив параметров, адаптированных алгоритмом обучения. Будем называть эти массивы тензорами. Поскольку каждый элемент входа и ядра должен храниться отдельно в явном виде, обычно предполагается, что эти функции равны нулю всюду, кроме конечного множества точек, для которых мы храним значения. На практике это означает, что сумму от минус до плюс бесконечности можно заменить суммированием по конечному числу элементов массива.

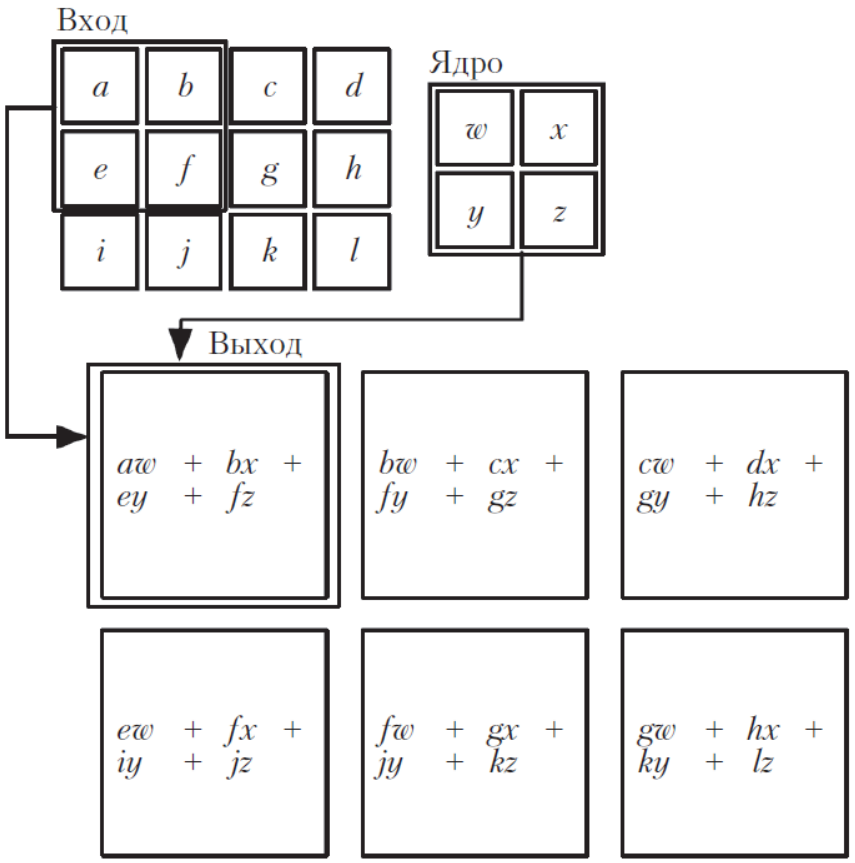
Во многих библиотеках машинного обучения реализована родственная функция – перекрестная корреляция – та же свертка, только без отражения ядра:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

Лекция 2.4 - Сверточные сети

МОТИВАЦИЯ

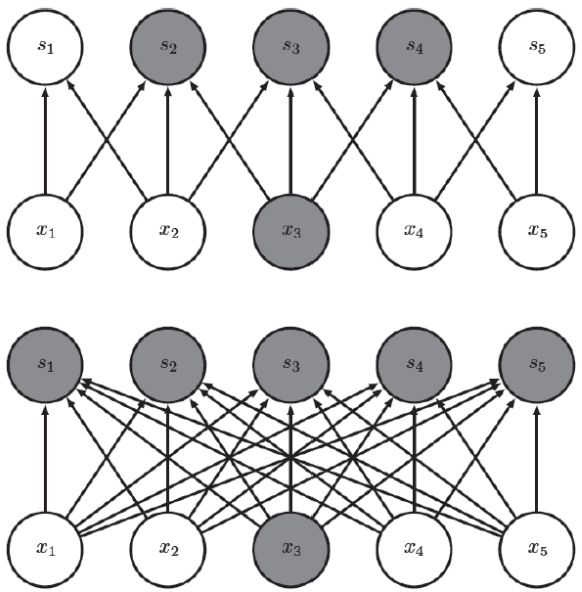
Со сверткой связаны три важные идеи, которые помогают улучшить систему машинного обучения: ***разреженные взаимодействия***, ***разделение параметров*** и ***эквивариантные представления***. Кроме того, свертка предоставляет средства для работы со входами переменного размера.



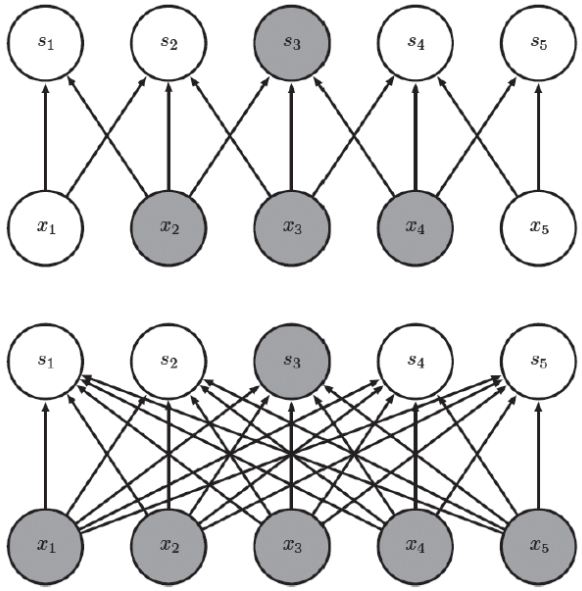
Пример двумерной свертки без отражения ядра. Выход ограничен только теми позициями, для которых ядро целиком укладывается в изображение; в некоторых контекстах такая свертка называется «допустимой». Прямоугольник с указывающими на него стрелками показывает, как левый верхний элемент выходного тензора образуется путем применения ядра к соответствующей левой верхней области входного тензора.

Лекция 2.4 - Сверточные сети

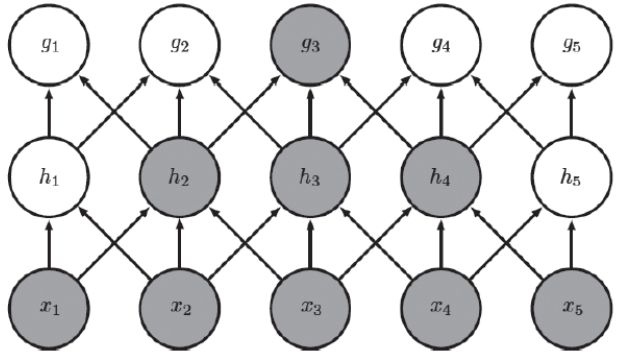
МОТИВАЦИЯ



Разреженная связность при взгляде снизу. Выделен один входной блок x_3 и выходные блоки в слое s, на которые этот блок влияет. (Вверху) Если s образован сверткой с ядром ширины 3, то x_3 влияет только на три выхода. (Внизу) Если s образован умножением на матрицу, то связность уже не разреженная, поэтому x3 влияет на все выходы.



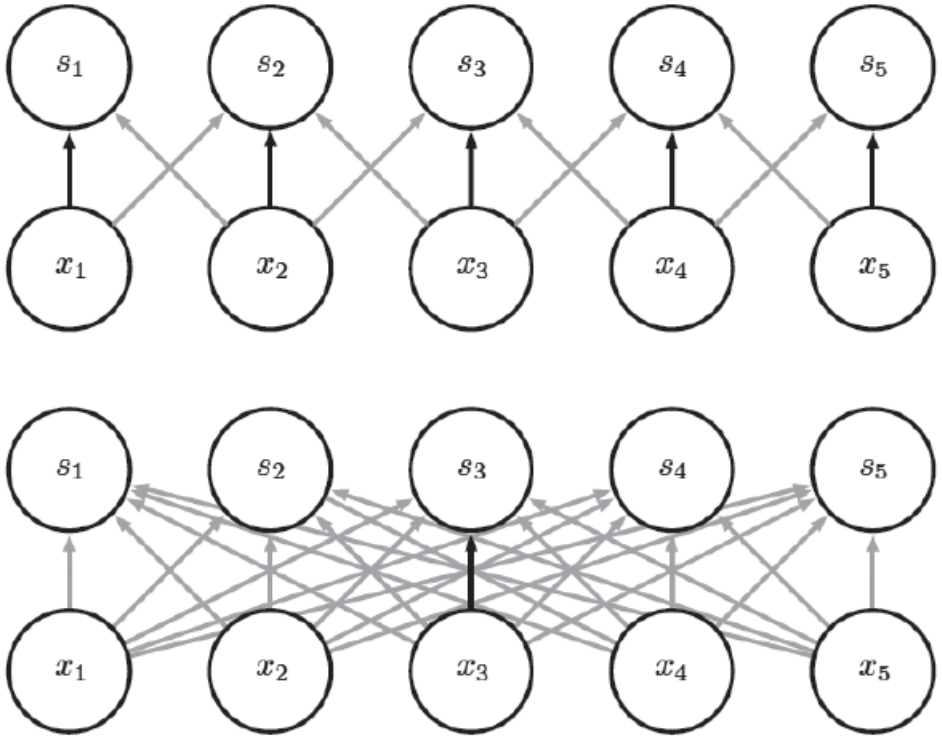
Разреженная связность при взгляде сверху. Выделен один выходной блок s_3 и выходные блоки в слое x, которые на него влияют. Совокупность этих блоков называется рецептивным полем s_3. (Вверху) Если s образован сверткой с ядром ширины 3, то на s_3 влияют только три входа. (Внизу) Если s образован умножением на матрицу, то связность уже не разреженная, поэтому на s_3 влияют все входы



Рецептивное поле блоков в глубоких слоях сверточной сети больше рецептивного поля в слоях, близких к поверхности. Этот эффект усиливается, если сеть включает такие архитектурные особенности, как свертка с шагом или пулинг. Это означает, что хотя прямые связи в сверточной сети действительно очень разрежены, блоки в глубоких слоях могут быть косвенно связаны со всем входным изображением или с большей его частью

Лекция 2.4 - Сверточные сети

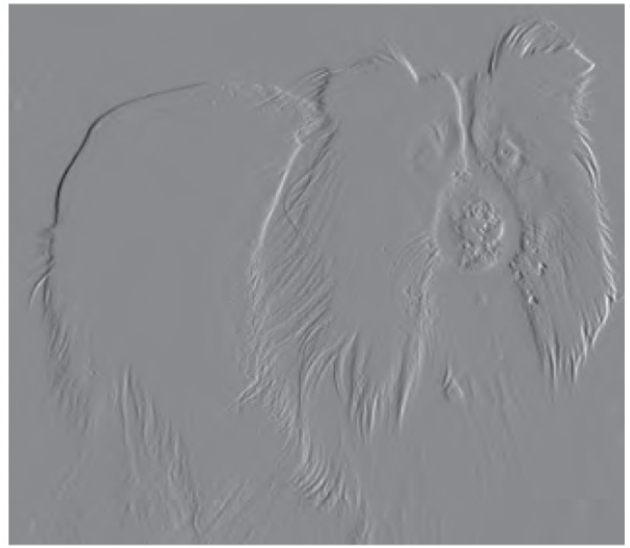
МОТИВАЦИЯ



Разделение параметров. Черными стрелками показаны связи, в которых участвует конкретный параметр в двух разных моделях. (Вверху) Черными стрелками показано использование центрального элемента 3-элементного ядра в сверточной модели. Благодаря разделению параметров этот параметр используется для всех элементов входа. (Внизу) Одиночная черная стрелка показывает использование центрального элемента матрицы весов в полносвязной модели. Здесь никакого разделения параметров нет, поэтому параметр используется только один раз.

Лекция 2.4 - Сверточные сети

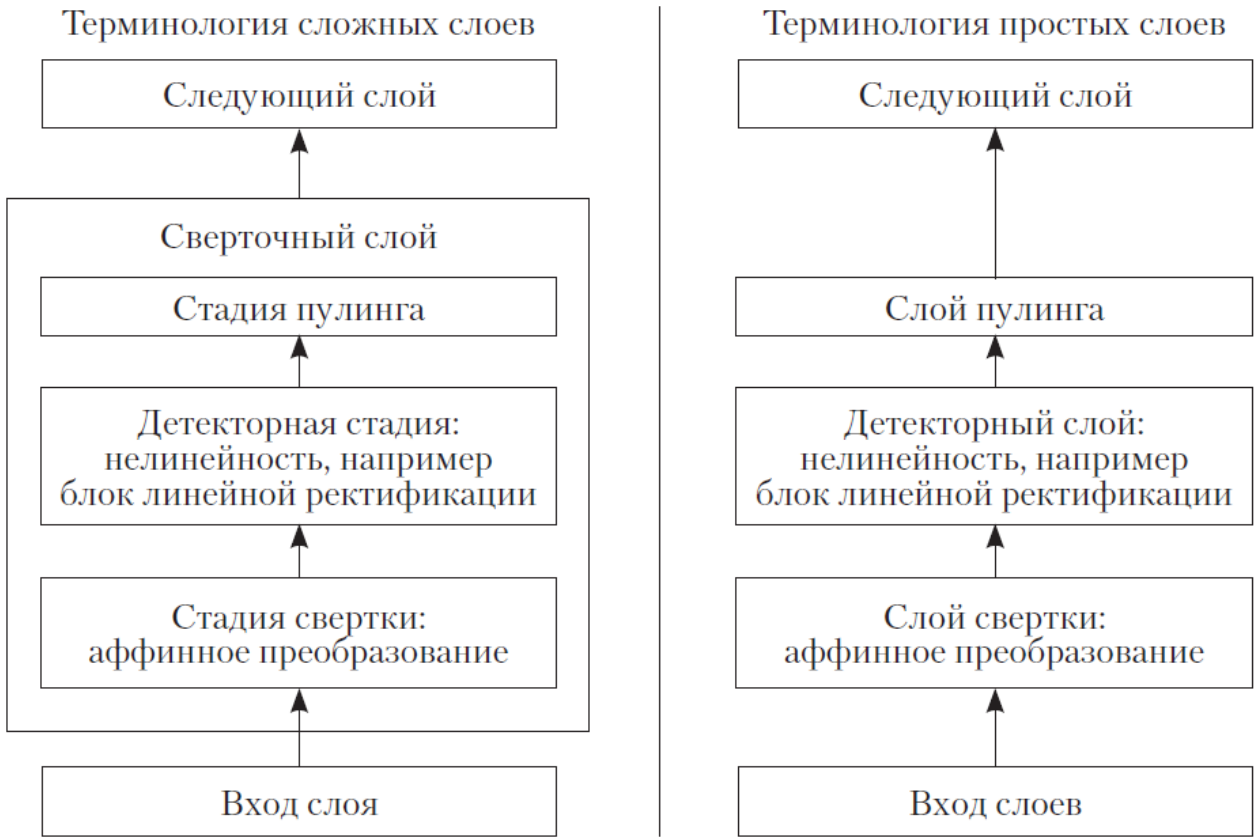
МОТИВАЦИЯ



Эффективность обнаружения границ. Правое изображение получено вычитанием из каждого пикселя исходного изображения значения пикселя слева от него. В результате мы получаем силу всех вертикальных границ во входном изображении, что бывает полезно для обнаружения объектов. Высота обоих изображений 280 пикселей. Ширина входного изображения 320 пикселей, а выходного – 319. Это преобразование можно описать как свертку с ядром, содержащим два элемента, оно требует 319 × 280 × 3 = 267 960 операций с плавающей точкой (два умножения и одно сложение на каждый выходной пиксель). Если то же самое преобразование выполнять путем перемножения матриц, то потребуется 320 × 280 × 319 × 280, т. е. больше восьми миллиардов элементов матрицы, так что с точки зрения потребления памяти свертка эффективнее такого преобразования в четыре миллиарда раз. При прямом перемножении матриц пришлось бы выполнить свыше 16 миллиардов операций с плавающей точкой, так что и с этой точки зрения свертка примерно в 60 000 раз эффективнее. Конечно, большинство элементов матрицы было бы равно нулю. Если хранить только ненулевые элементы, то в обоих случаях пришлось бы выполнить примерно одно и то же число операций с плавающей точкой. Но все равно в матрице было бы 2 × 319 × 280 = 178 640 элементов. Свертка – чрезвычайно эффективный способ описания преобразований, в которых одно и то же линейное преобразование многократно применяется к небольшим участкам изображения

Лекция 2.4 - Сверточные сети

ПУЛИНГ

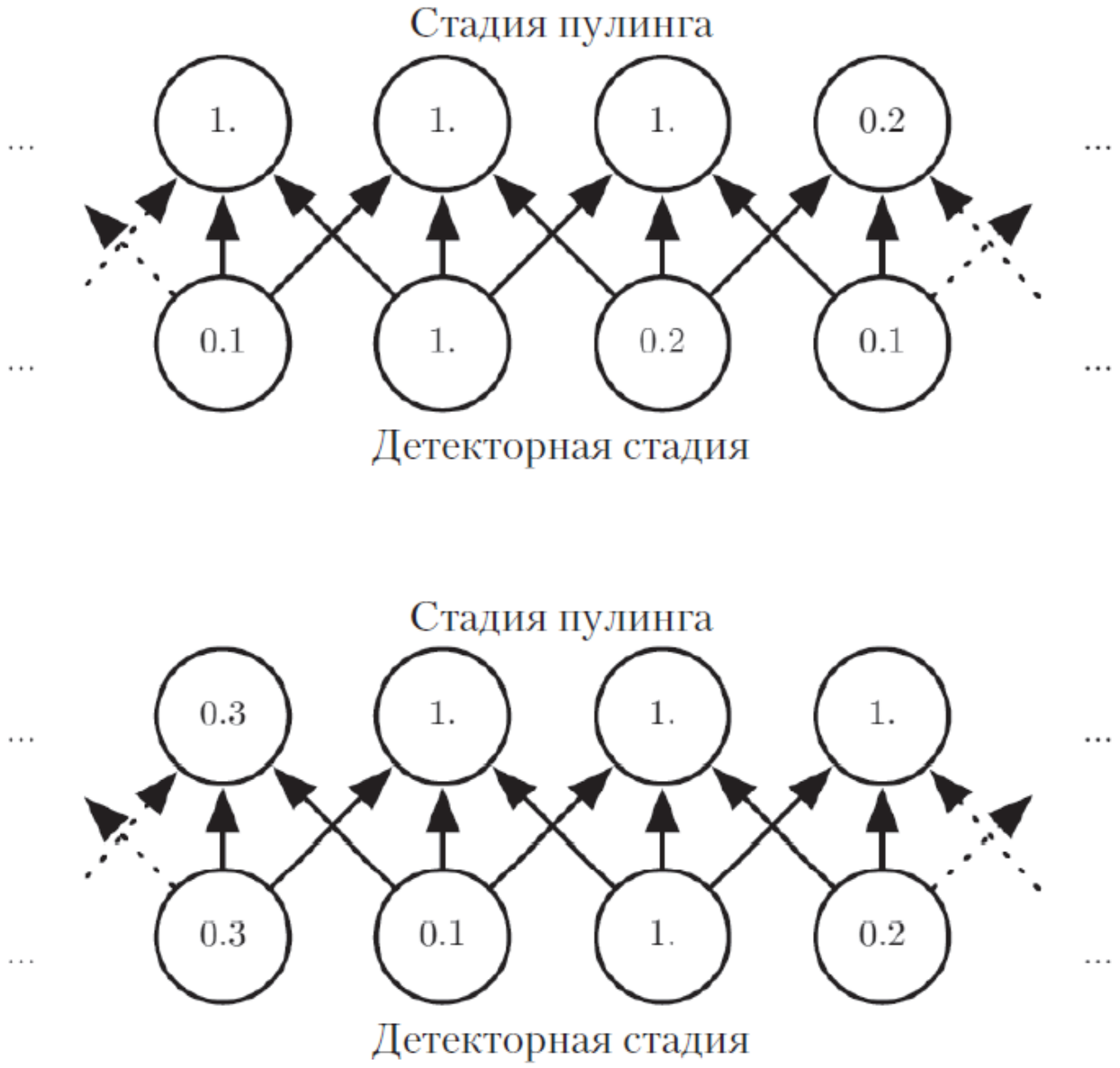


Компоненты типичного слоя сверточной нейронной сети. Для описания таких слоев применяется двоякая терминология. (Слева) В этой терминологии сверточная сеть рассматривается как небольшой набор относительно сложных слоев, каждый из которых имеет много «стадий». При этом существует взаимно однозначное соответствие между ядерными тензорами и слоями сети. В этой книге мы в основном придерживаемся такой терминологии. (Справа) В этой терминологии сверточная сеть рассматривается как большой набор простых слоев слоев, а каждый шаг обработки считается полноправным слоем. Это означает, что не у каждого «слоя» есть параметры

Лекция 2.4 - Сверточные сети

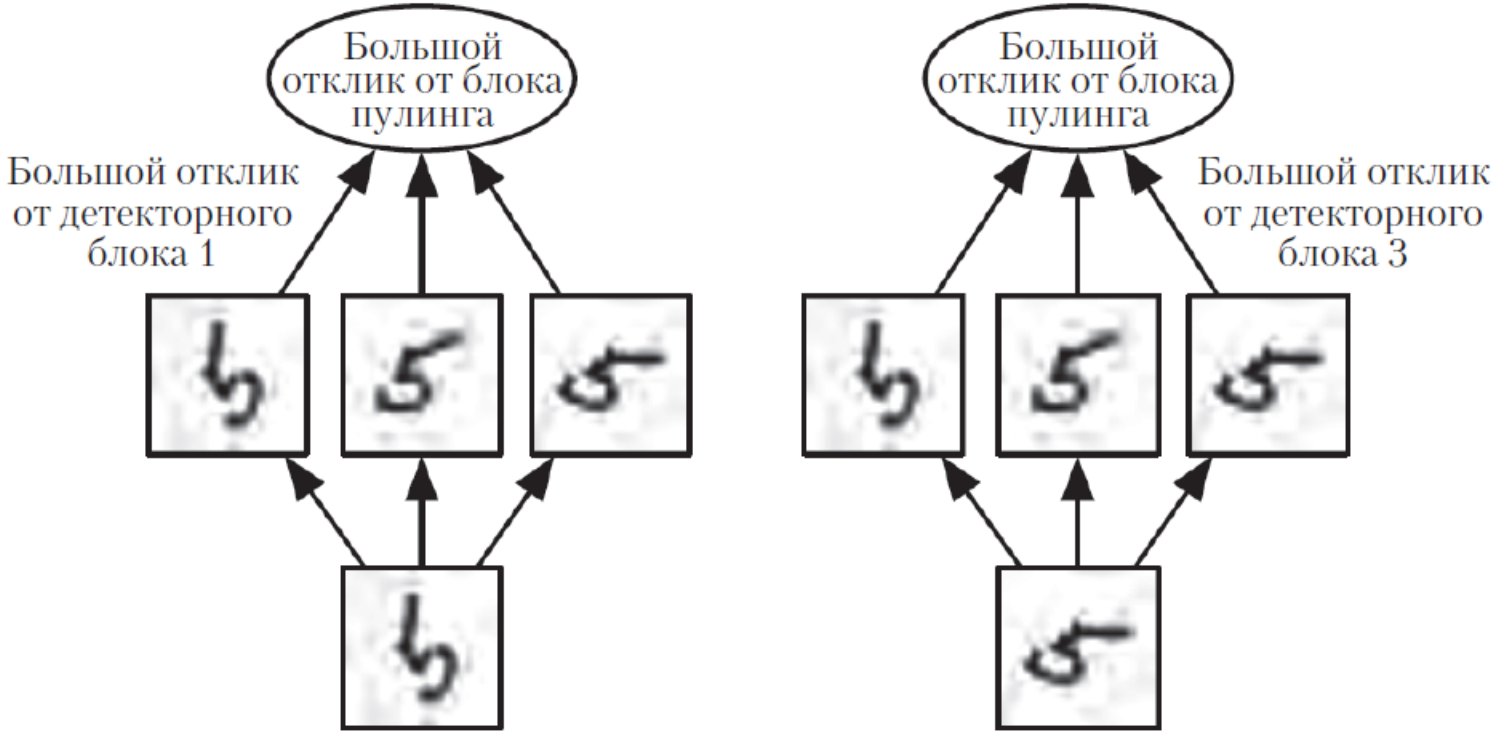
ПУЛИНГ

Max-пулинг привносит инвариантность. (Вверху) Середина выходного слоя сверточной сети. В нижней строке показаны выходы нелинейности, а в верхней – выходы max-пулинга с шагом в один пиксель между областями пулинга, каждая из которых имеет ширину три пикселя. (Вверху) Та же самая сеть, сдвинутая вправо на один пиксель. В нижней строке изменились все значения, а в верхней – только половина, потому что блоки max-пулинга чувствительны лишь к максимальному значению в своей окрестности, а не точному положению этого значения



Лекция 2.4 - Сверточные сети

ПУЛИНГ

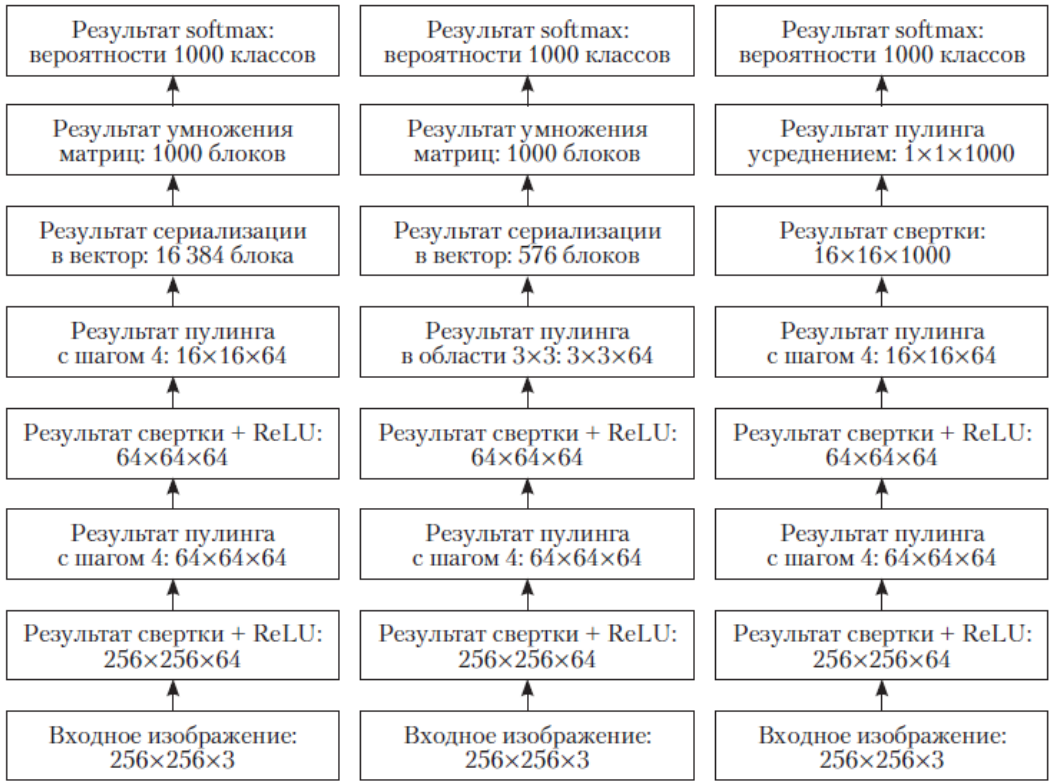


Пример обученной инвариантности. Блок, выполняющий пулинг по нескольким признакам, обученным с разными параметрами, может обучиться инвариантности к преобразованиям входа. Здесь мы видим набор из трех обученных фильтров и блок max-пулинга, который обучился инвариантности к вращению. Все три фильтра предназначены для распознавания рукописной цифры 5. Каждый фильтр настроен на свою ориентацию пятерки. Если на входе появляется цифра 5, то соответствующий фильтр распознает ее, что даст большой отклик на детекторный блок. Тогда блок max-пулинга даст большой отклик независимо от того, какой детекторный блок был активирован. На рисунке показано, как сеть обрабатывает два разных входа, активирующих разные детекторные блоки. В обоих случаях выход блока пулинга примерно одинаков. Max-пулинг по пространственной области обладает естественной инвариантностью к параллельным переносам; такой многоканальный подход необходим только для обучения другим преобразованиям.

Лекция 2.4 - Сверточные сети

СВЕРТКА И ПУЛИНГ КАК БЕСКОНЕЧНО СИЛЬНОЕ АПРИОРНОЕ РАСПРЕДЕЛЕНИЕ

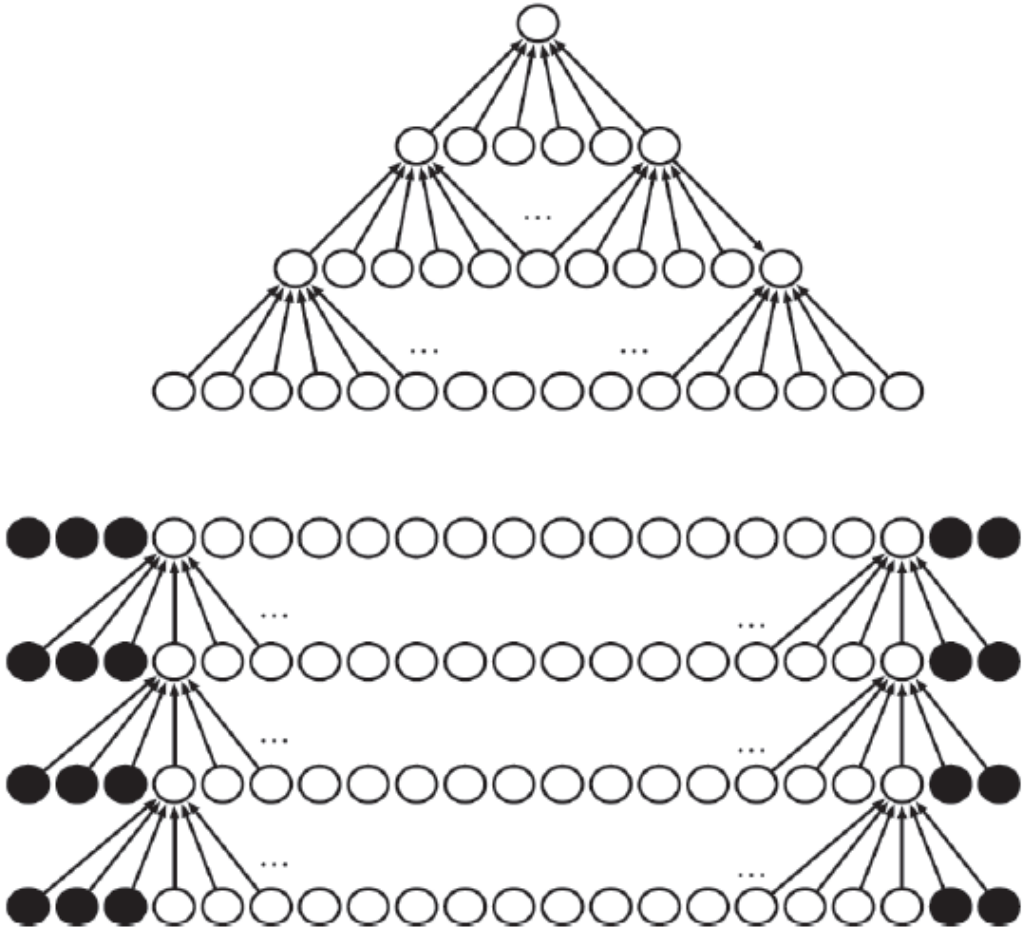
Примеры архитектур для классификации на основе сверточных сетей. Конкретные величины шага и глубины на этом рисунке не следует рассматривать как рекомендации для практического применения; они выбраны так, чтобы рисунок поместился на страницу. Кроме того, реальные сверточные сети часто являются сильно разветвленными, а не цепными, как на этом рисунке. (Слева) Сверточная сеть для обработки изображений фиксированного размера. После чередования нескольких сверточных и пулинговых слоев форма тензора сверточной карты признаков меняется, чтобы выстроить все пространственные измерения в один плоский вектор. Далее следует обычный классификатор в виде сети прямого распространения, описанный в главе 6. (В центре) Сверточная сеть для обработки изображений переменного размера, в которой по-прежнему имеется полносвязный участок. В сети применяется операция пулинга с фиксированным числом пулов переменного размера, которая порождает вектор из 576 блоков, подаваемый на вход полносвязного участка сети. (Справа) Сверточная сеть, в которой вообще нет полносвязного слоя весов. Вместо этого последний сверточный слой выводит по одной карте признаков на каждый класс. Предположительно модель обучает карту тому, насколько вероятно появление каждого класса в каждой пространственной области. Единственное значение, получающееся в результате усреднения карты признаков, подается на вход softmax-классификатора в верхнем слое.



Лекция 2.4 - Сверточные сети

ВАРИАНТЫ БАЗОВОЙ ФУНКЦИИ СВЕРТКИ

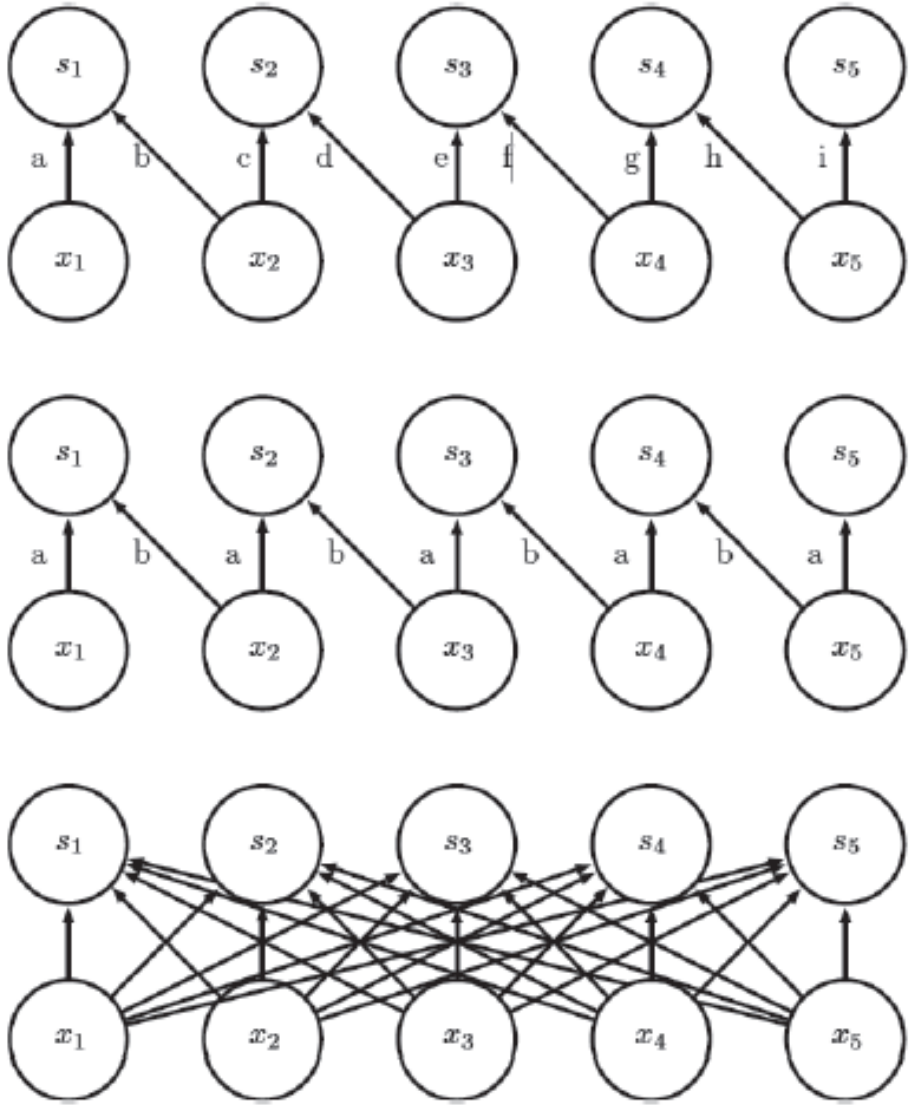
Влияние дополнения нулями на размер сети. Рассматривается сверточная сеть с ядром ширины 6 в каждом слое. В этом примере пулинг не используется, поэтому уменьшение размера сети вызвано только самой операцией свертки. (Вверху) В этой сверточной сети нет никакого неявного дополнения нулями. Поэтому ширина представления уменьшается на пять пикселей с каждым слоем. Если ширина входа равна 16 пикселям, то в сети может быть не более трех сверточных слоев, причем по последнему переместить ядро невозможно, так что лишь два слоя можно назвать сверточными. Скорость сжатия можно уменьшить, если использовать ядра поменьше, но малые ядра обладают меньшей выразительностью, а сжатие как таковое все равно присутствует. (Снизу) Неявно добавив пять нулей в каждый слой, мы препятствуем сжатию представления. В результате глубина сверточной сети может быть любой.



Лекция 2.4 - Сверточные сети

ВАРИАНТЫ БАЗОВОЙ ФУНКЦИИ СВЕРТКИ

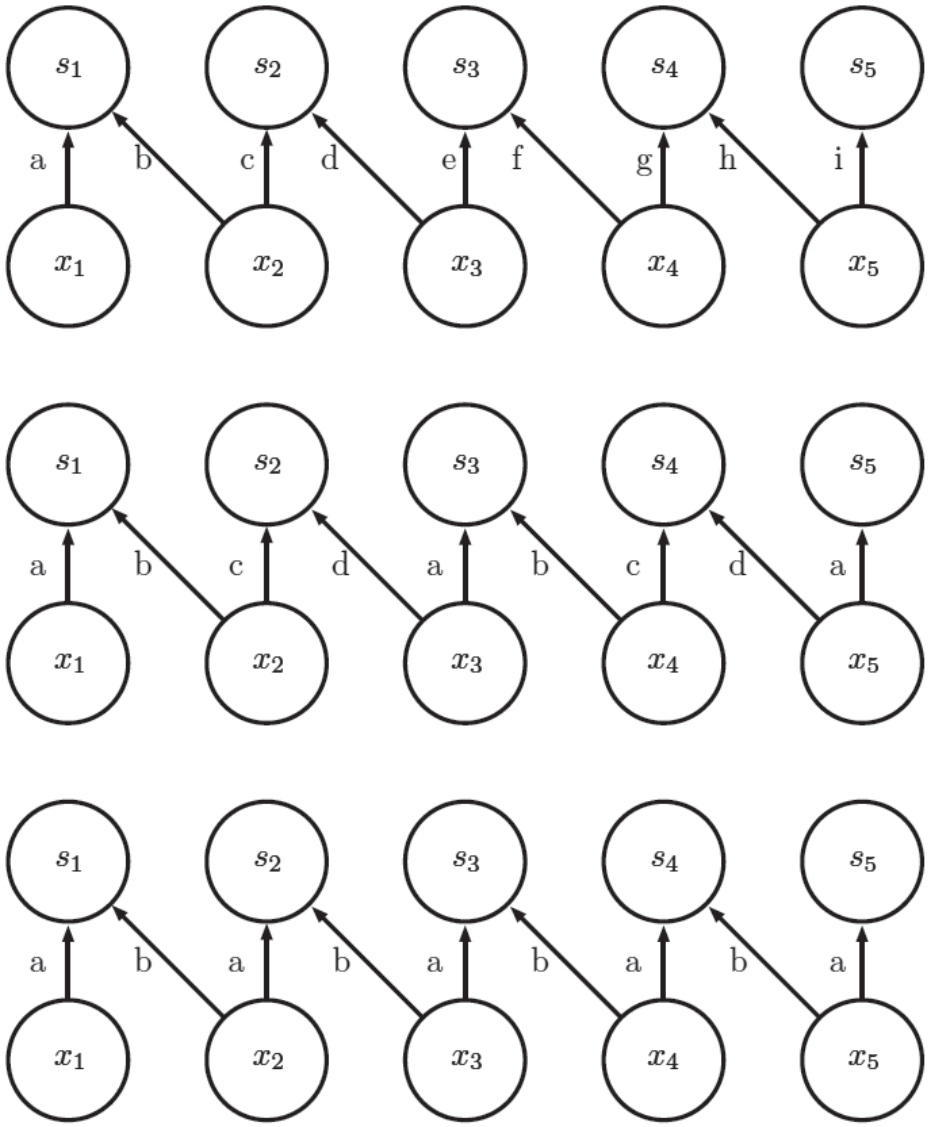
Сравнение локальной связности, свертки и полной связности. (Вверху) Локально связный слой с патчем шириной два пикселя. Все ребра помечены разными буквами, т. е. с каждым ребром ассоциирован свой весовой параметр. (В центре) Сверточный слой с ядром шириной два пикселя. Связность этой модели точно такая же, как у локально связного слоя. Различие не в том, какие блоки взаимодействуют друг с другом, а в том, как разделяются параметры. В локальном связном слое никакого разделения параметров нет. В сверточном слое два одинаковых же веса повторно используются для всего входного слоя, это видно из повторения буквенных меток ребер. (Внизу) Полносвязный слой похож на локально связный в том смысле, что у каждого ребра свой собственный параметр (их слишком много, чтобы расставлять буквы на рисунке). А отличие в том, что в локальном связном слое связность ограничена



Лекция 2.4 - Сверточные сети

ВАРИАНТЫ БАЗОВОЙ ФУНКЦИИ СВЕРТКИ

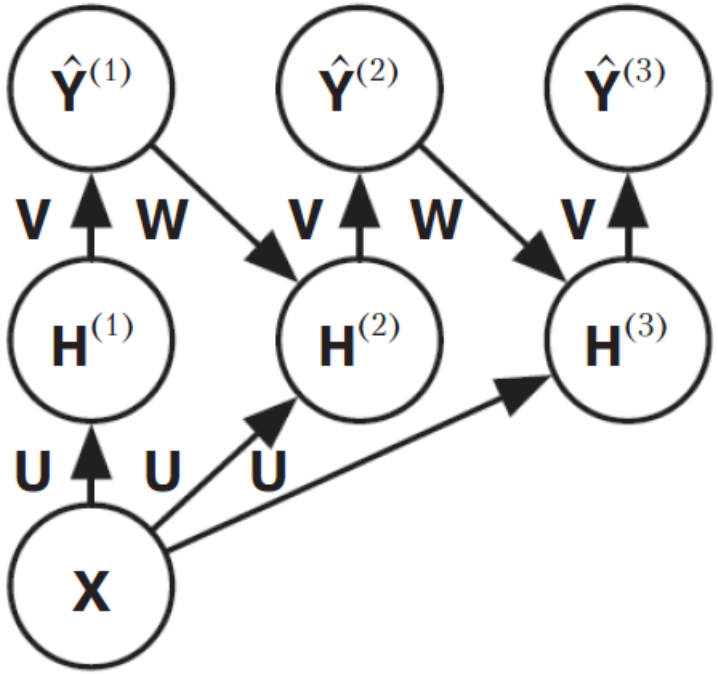
Сравнение локально связных слоев, периодической свертки и стандартной свертки. Во всех трех случаях при использовании ядра одного размера набор связей между блоками один и тот же. На этом рисунке предполагается, что ширина ядра составляет два пикселя. Различие между методами – в разделении параметров. (Вверху) В локально связном слое параметры не разделяются вовсе. Все связи помечены разными буквами, т. е. у каждой связи свой вес. (В центре) В случае периодической свертки имеется набор из t разных ядер. В данном случае t = 2. Ребра первого ядра помечены буквами «a» и «b», а ребра второго – буквами «c» и «d». При смещении в выходном слое на один пиксель вправо мы переходим к использованию другого ядра. Это означает, что, как и в локально связном слое, у соседних выходных блоков параметры разные. Но, в отличие от локально связного слоя, после перебора всех t имеющихся ядер мы снова возвращаемся к первому. Два выходных блока, разделенных числом шагов, кратным t, разделяют общие параметры. (Внизу) Традиционная свертка эквивалентна периодической с t = 1. Существует всего одно ядро, которое применяется во всех точках. На рисунке это следует из того, что все ребра помечены буквами «a» и «b»



Лекция 2.4 - Сверточные сети

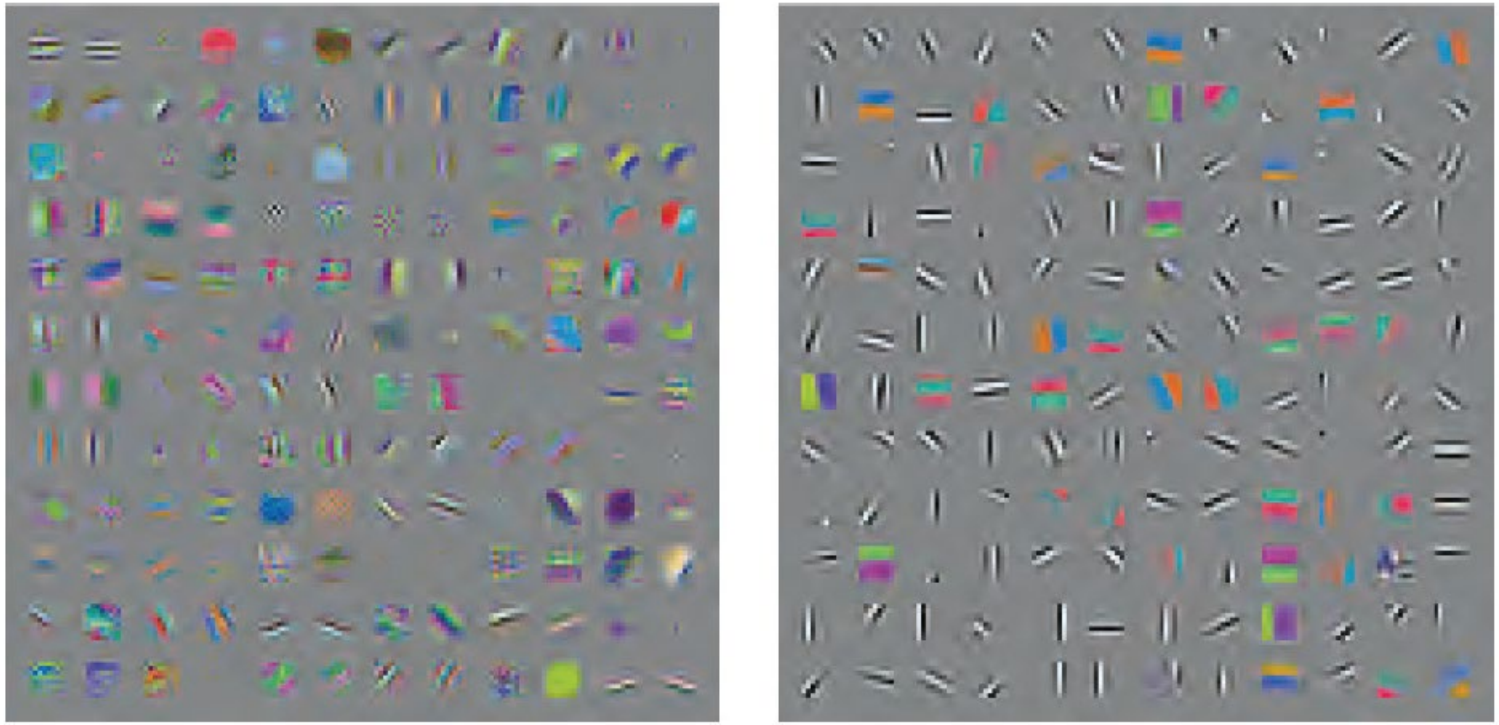
СТРУКТУРИРОВАННЫЙ ВЫХОД

Пример рекуррентной сверточной сети для пометки пикселей. Входом является тензор изображения X, оси которого соответствуют строкам, столбцам и каналам (красный, зеленый, синий) изображения. Цель – построить тензор меток Yˆ, содержащий распределение вероятности меток каждого пикселя. Оси этого тензора соответствуют строкам, столбцам и классам. Вместо того чтобы выводить Yˆ за один присест, рекуррентная сеть итеративно уточняет оценку Yˆ, используя предыдущую оценку как исходные данные для вычисления новой. Для каждого приближения к оценке используются одни и те же параметры, количество уточнений может быть любым. На каждом шаге используется тензор сверточных ядер U для вычисления скрытого представления входного изображения. Ядерный тензор V используется для порождения оценки меток при заданных скрытых значениях. На всех шагах, кроме первого, ядра W сворачиваются с Yˆ для вычисления входа скрытого слоя. На первом шаге этот член заменяется нулем.



Лекция 2.4 - Сверточные сети

РАЗРЕЖЕННОЕ КОДИРОВАНИЕ

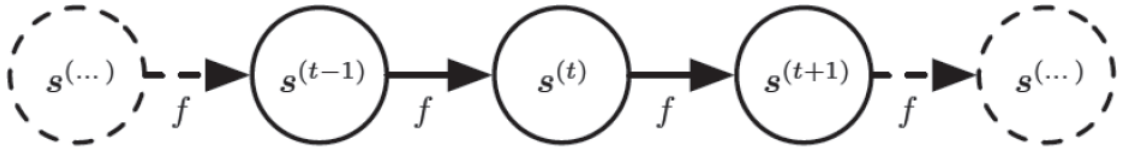


Многие алгоритмы машинного обучения обучают признаки для обнаружения любых границ или границ определенного цвета в естественных изображениях. Такие детекторы напоминают функции Габора, принимающие участие в работе первичной зрительной коры. (Слева) Веса, обученные алгоритмом обучения без учителя (разреженное кодирование типа Spike-and-Slab) в применении к небольшим участкам изображения. (Справа) Сверточные ядра, обученные с учителем первым слоем сверточной maxout-сети. Соседние пары фильтров управляют одним и тем же maxout-блоком

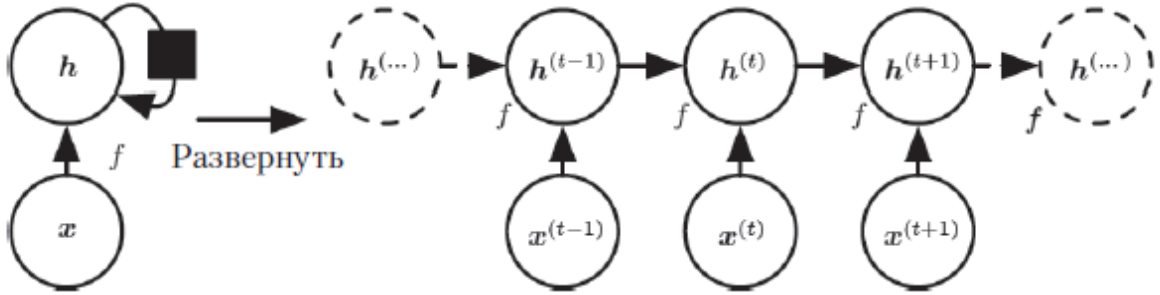
Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

Рекуррентные нейронные сети, или РНС, – это семейство нейронных сетей для обработки последовательных данных.

РАЗВЕРТКА ГРАФА ВЫЧИСЛЕНИЙ



Классическая динамическая система, описываемая выражением $s^{(t)} = f(s^{(t-1)}; \theta)$, в виде развернутого графа вычислений. Каждая вершина представляет состояние в некоторый момент t, а функция f отображает состояние в момент t на состояние в момент t + 1. Одни и те же параметры (значение θ, параметризующее f) используются на всех временных шагах.



Рекуррентная сеть без выходов. Эта сеть просто обрабатывает информацию из входа x, включая ее в состояние h, которое передается дальше во времени. (Слева) Принципиальная схема. Черный квадратик обозначает задержку на один временной шаг. (Справа) Та же сеть в виде развернутого графа вычислений, в котором каждая вершина ассоциирована с одним моментом времени

Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

РАЗВЕРТКА ГРАФА ВЫЧИСЛЕНИЙ

Мы можем представить развернутое рекуррентное выражение после t шагов функцией $g^{(t)}$:

$$\begin{aligned} \boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}). \end{aligned}$$

Таким образом, процесс развертки дает два важных преимущества:

- независимо от длины последовательности размер входа обученной модели всегда один и тот же, поскольку он описывается в терминах перехода из одного состояния в другое, а не в терминах истории состояний переменной длины;
- одну и ту же функцию перехода f с одними и теми же параметрами можно использовать на каждом шаге.

Вооружившись механизмами развертки графов и разделения параметров, мы можем перейти к проектированию разнообразных рекуррентных нейронных сетей. Вот несколько важных паттернов проектирования таких сетей:

- Рекуррентные сети, порождающие выход на каждом временном шаге и имеющие рекуррентные связи между скрытыми блоками;
- Рекуррентные сети, порождающие выход на каждом временном шаге и имеющие рекуррентные связи только между выходами на одном временном шаге и скрытыми блоками на следующем;
- Рекуррентные сети с рекуррентными связями между скрытыми блоками, которые читают последовательность целиком, а затем порождают единственный выход

Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

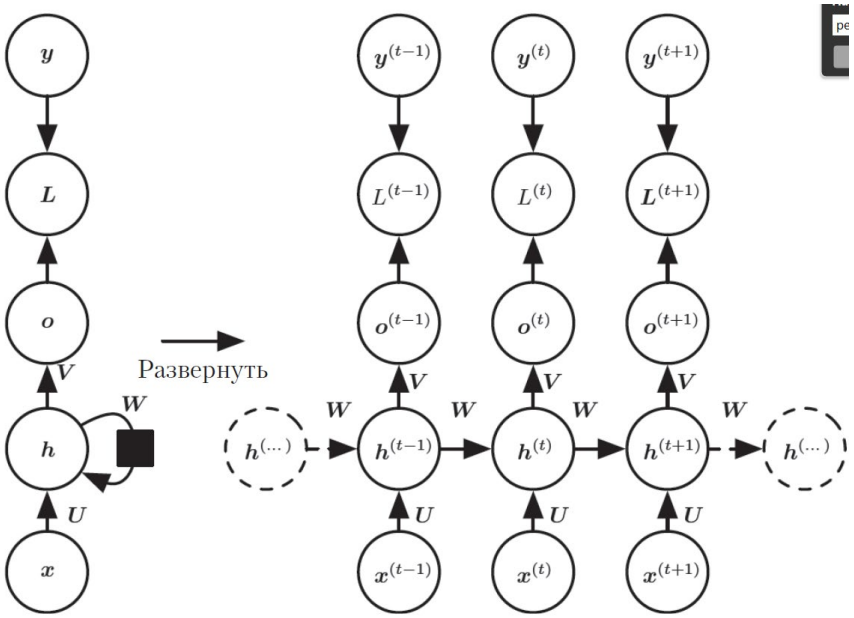
Прямое распространение начинается с задания начального состояния $h^{(0)}$. Затем для каждого временного шага от t = 1 до t = τ применяем следующие уравнения обновления:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$

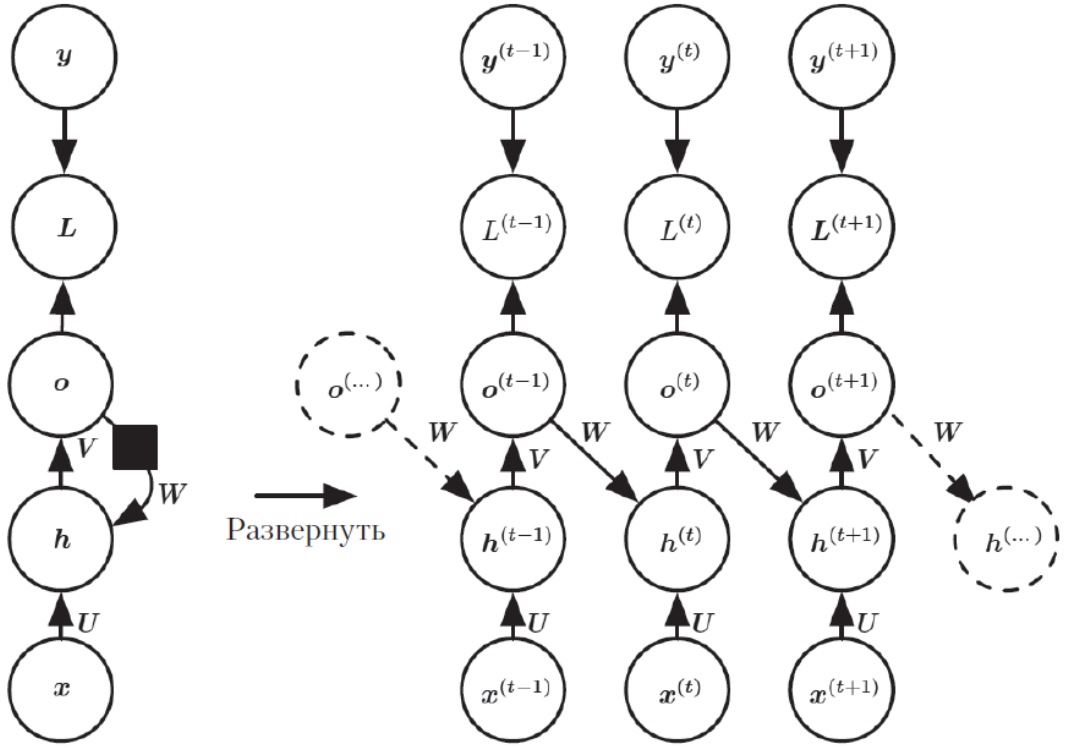
$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)}),$$



Граф вычислений потерь при обучении рекуррентной сети, которая отображает входную последовательность значений x в соответствующую выходную последовательность значений o. Функция потерь L измеряет, насколько далеко каждый элемент o отстоит от соответствующей метки y. В случае применения к выходам функции softmax можно предполагать, что o – ненормированные логарифмические вероятности. Внутри функция L вычисляет yˆ = softmax(o) и сравнивает эту величину с меткой y. В РНС имеются связи между входным и скрытым слоями, параметризованные матрицей весов U, рекуррентные связи между скрытыми блоками, параметризованные матрицей весов W, и связи между скрытым и выходным слоями, параметризованные матрицей весов V. (Слева) РНС и ее функция потерь, представленные в виде рекуррентных связей. (Справа) То же в виде развернутого во времени графа вычислений, в котором каждая вершина ассоциирована с одним моментом времени

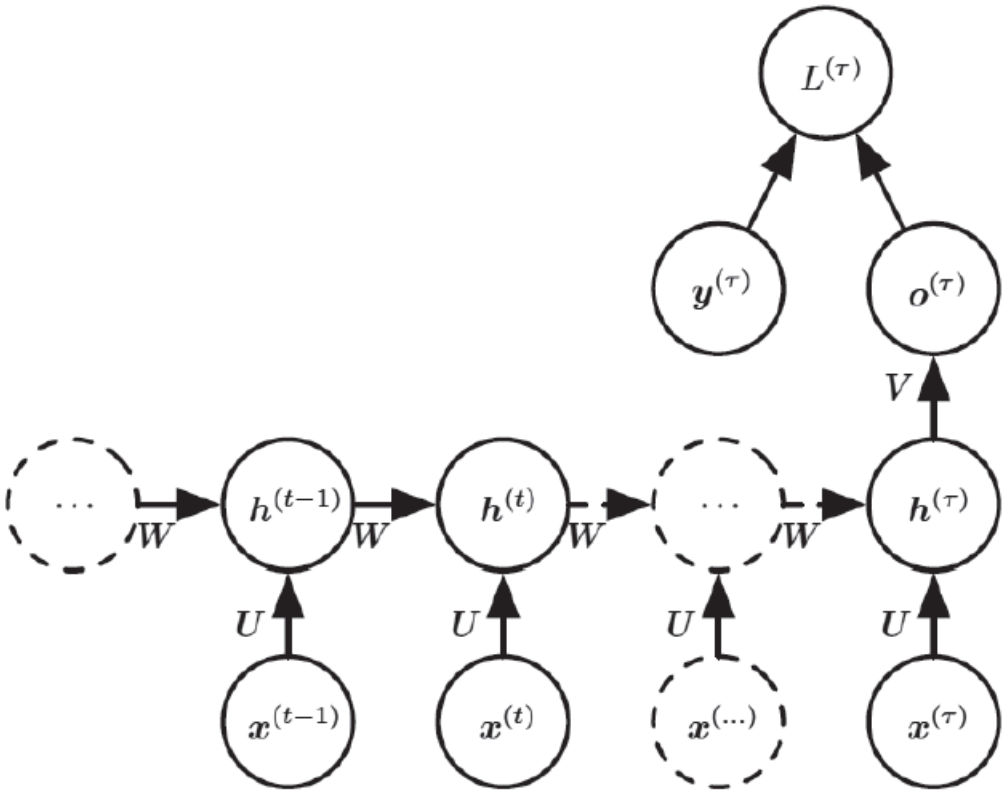
Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ



РНС, в которой единственным видом рекурсии является обратная связь между выходным и скрытым слоями. На каждом временном шаге t вход обозначен x^t, активации скрытого слоя – $h^{(t)}$, выходы – $o^{(t)}$, метки – $y^{(t)}$, а потеря – $L^{(t)}$. (Слева) Принципиальная схема. (Справа) Развернутый граф вычислений. РНС на этом рисунке обучена помещать конкретное выходное значение в o, и o – единственная информация, которую разрешено передавать в будущее. Не существует прямых горизонтальных связей, исходящих из h. Предыдущий h связан с настоящим только косвенно – с помощью порожденных им предсказаний. Если размерность вектора o не слишком велика, то в нем обычно отсутствует какая-то важная информация о прошлом. Поэтому такая РНС менее мощная, но зато ее легче обучить, потому что каждый временной шаг можно обучать отдельно от всех остальных, благодаря чему возможно в большей степени распараллелить обучение

РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ



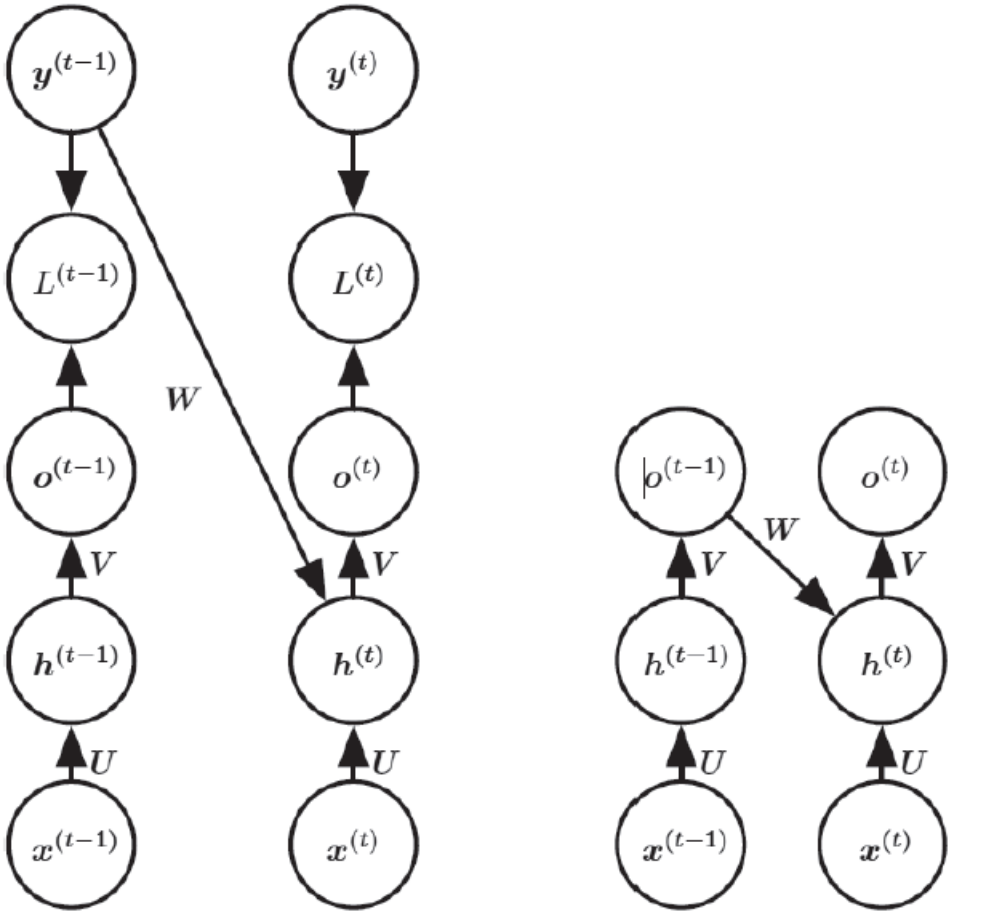
Развернутая во времени рекуррентная нейронная сеть с единственным выходом в конце последовательности. Такую сеть можно использовать для агрегирования последовательности и порождения представления фиксированного размера, подаваемого на вход следующего этапа обработки. В самом конце может быть сравнение с меткой (как показано на рисунке), но можно также получить градиент $o^{(t)}$ посредством обратного распространения от последующих модулей

Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

ФОРСИРОВАНИЕ УЧИТЕЛЯ И СЕТИ С РЕКУРСИЕЙ НА ВЫХОДЕ

Критерий условного максимального правдоподобия имеет вид:

$$\log p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$$
$$= \log p(\boldsymbol{y}^{(2)} \mid \boldsymbol{y}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) + \log p(\boldsymbol{y}^{(1)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$$

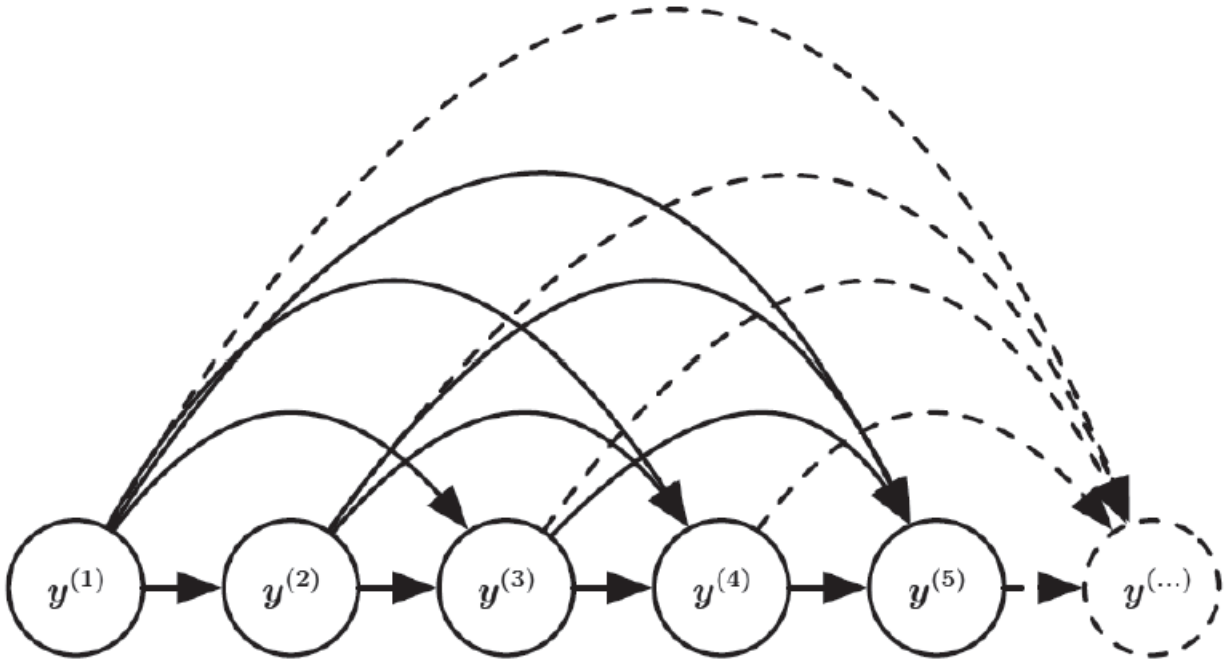


На этапе обучения

На этапе тестирования

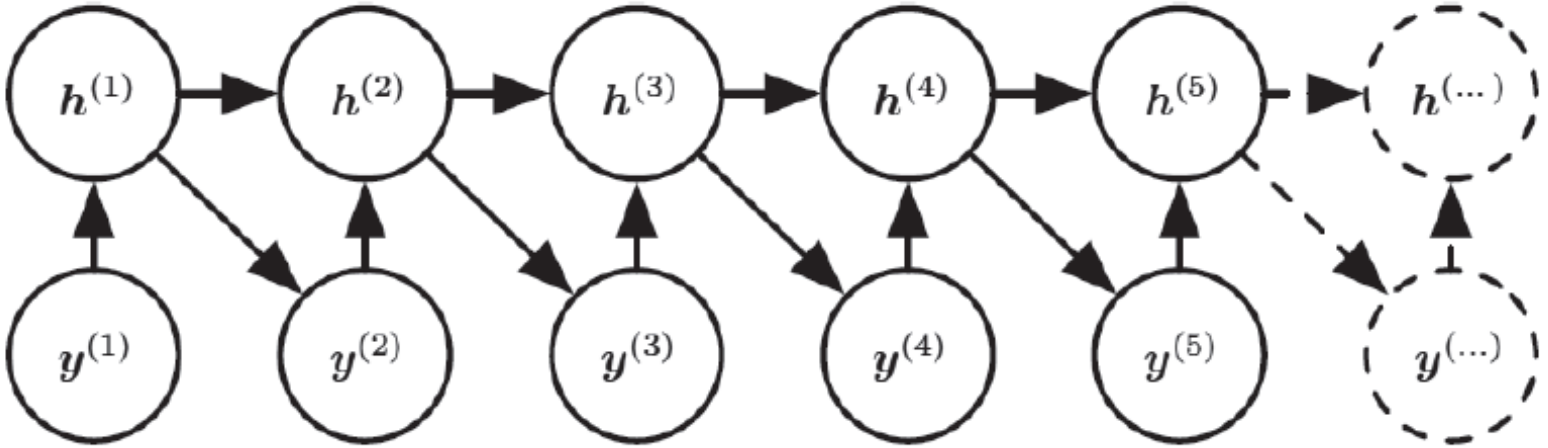
Форсирование учителя – метод обучения, применимый к РНС, в которых есть связи между выходом и скрытым состоянием на следующем временном шаге. (Слева) На этапе обучения мы подаем истинный выход $y^{(t)}$, взятый из обучающего набора, на вход $h^{(t+1)}$. (Справа) После того как модель развернута, истинный выход, вообще говоря, неизвестен. В таком случае мы аппроксимируем истинный выход $y^{(t)}$ выходом модели $o^{(t)}$ и подаем этот выход обратно в модель

РЕКУРРЕНТНЫЕ СЕТИ КАК ОРИЕНТИРОВАННЫЕ ГРАФИЧЕСКИЕ МОДЕЛИ



Полносвязная графическая модель последовательности $y^{(1)}$, $y^{(1)}$, …, $y^{(t)}$, …. Любое прошлое наблюдение $y^{(i)}$ может повлиять на условное распределение некоторых $y^{(t)}$ (для $t > i$) при условии предыдущих значений. Параметризация графической модели в точном соответствии с эти графом может оказаться крайне неэффективной, поскольку число входов и параметров для каждого следующего элемента последовательности будет постоянно возрастать.

РЕКУРРЕНТНЫЕ СЕТИ КАК ОРИЕНТИРОВАННЫЕ ГРАФИЧЕСКИЕ МОДЕЛИ



Введение переменной состояния в графическую модель РНС, хотя она и является детерминированной функцией своих аргументов, помогает понять, как можно получить очень эффективную параметризацию. Все участки этой последовательности (содержащие $h^{(t)}$ и $y^{(t)}$) имеют одинаковую структуру (одно и то же число входов для каждой вершины) и могут разделять параметры с другими участками.

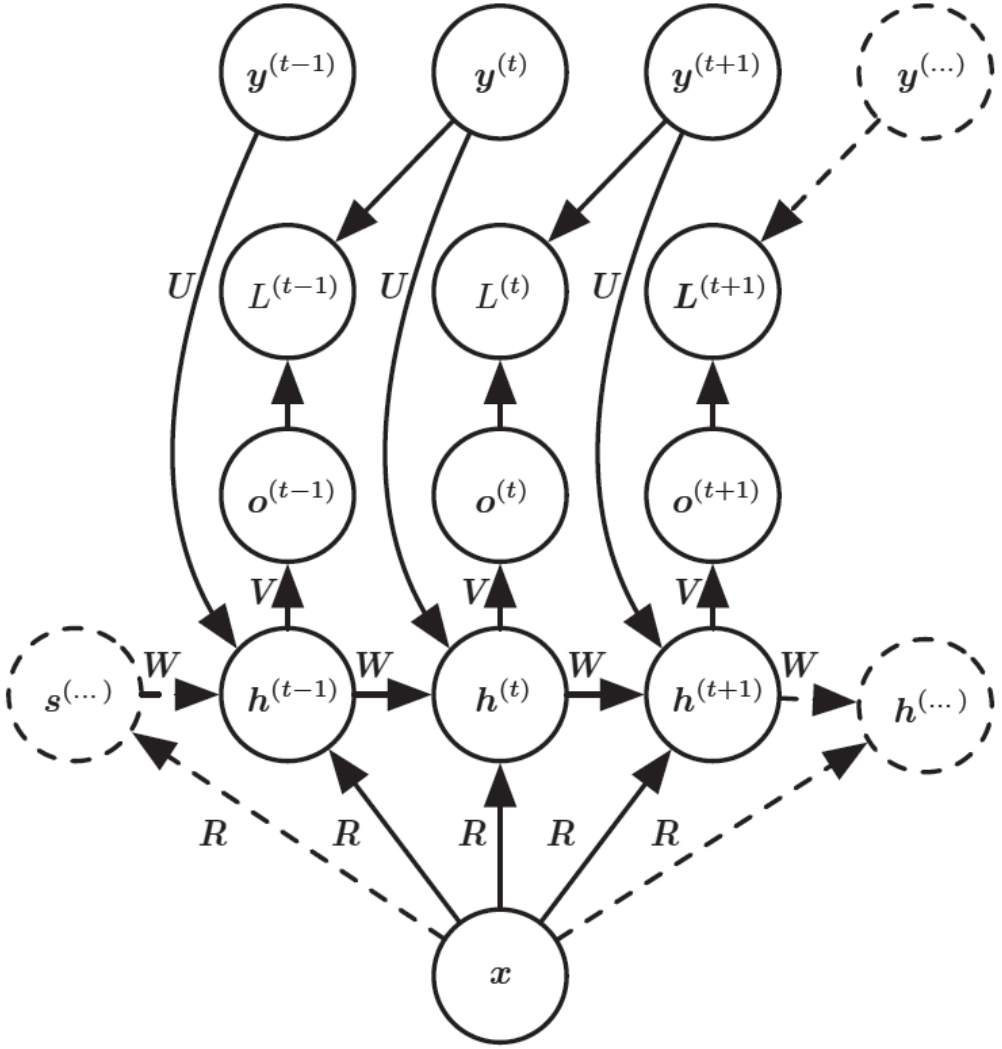
Лекция 2.5 – Моделирование рекуррентных и рекурсивных сетей

МОДЕЛИРОВАНИЕ КОНТЕКСТНО-ОБУСЛОВЛЕННЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ С ПОМОЩЬЮ РНС

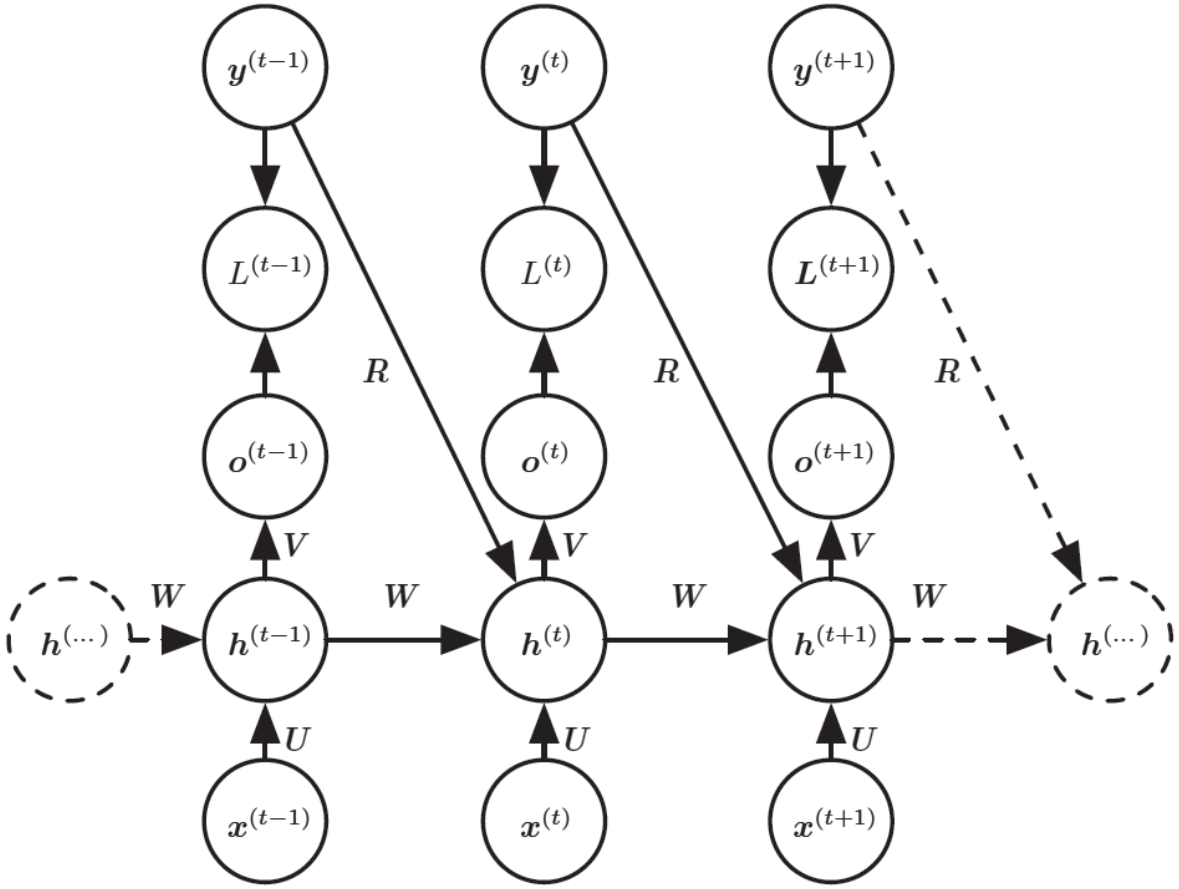
Вот несколько типичных способов подать дополнительный вход РНС:

- в качестве дополнительного входа на каждом временном шаге;
- в качестве начального состояния $h^{(0)}$;
- то и другое.

РНС, отображающая вектор фиксированной длины x в распределение последовательностей Y. Эта РНС подходит для таких задач, как подписывание изображений, когда одно изображение подается на вход модели, порождающей его описание в виде последовательности слов. Каждый элемент $y^{(t)}$ наблюдаемой выходной последовательности служит одновременно входом (для текущего временного шага) и – на этапе обучения – меткой (для предыдущего временного шага)



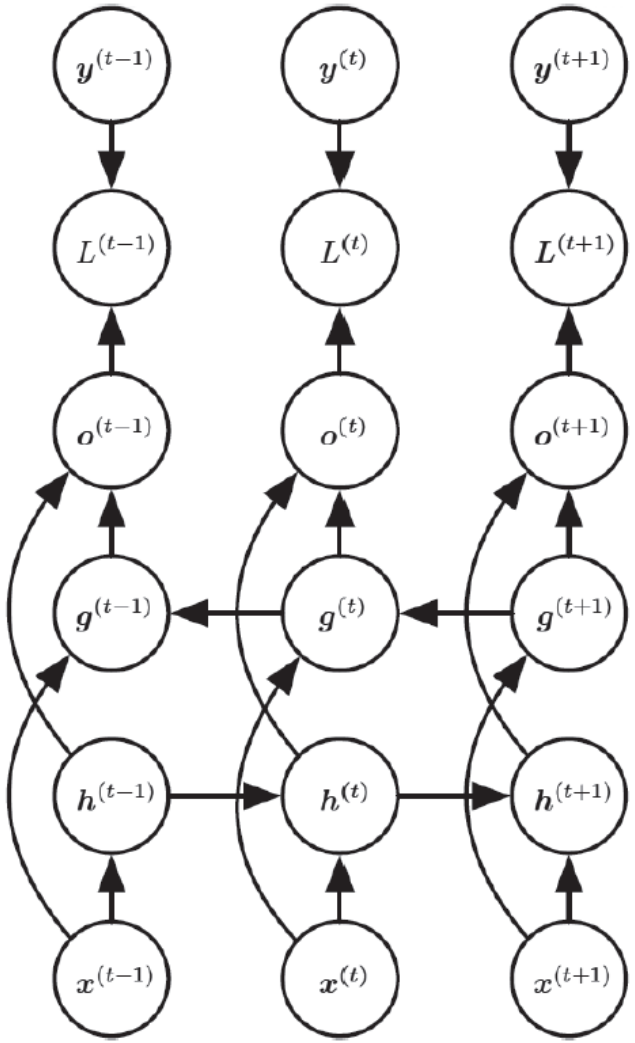
МОДЕЛИРОВАНИЕ КОНТЕКСТНО-ОБУСЛОВЛЕННЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ С ПОМОЩЬЮ РНС



Условная рекуррентная нейронная сеть, отображающая последовательность значений x переменной длины в распределение последовательностей значений у такой же длины.

АРХИТЕКТУРЫ КОДИРОВЩИК-ДЕКОДЕР ИЛИ ПОСЛЕДОВАТЕЛЬНОСТЬ В ПОСЛЕДОВАТЕЛЬНОСТЬ

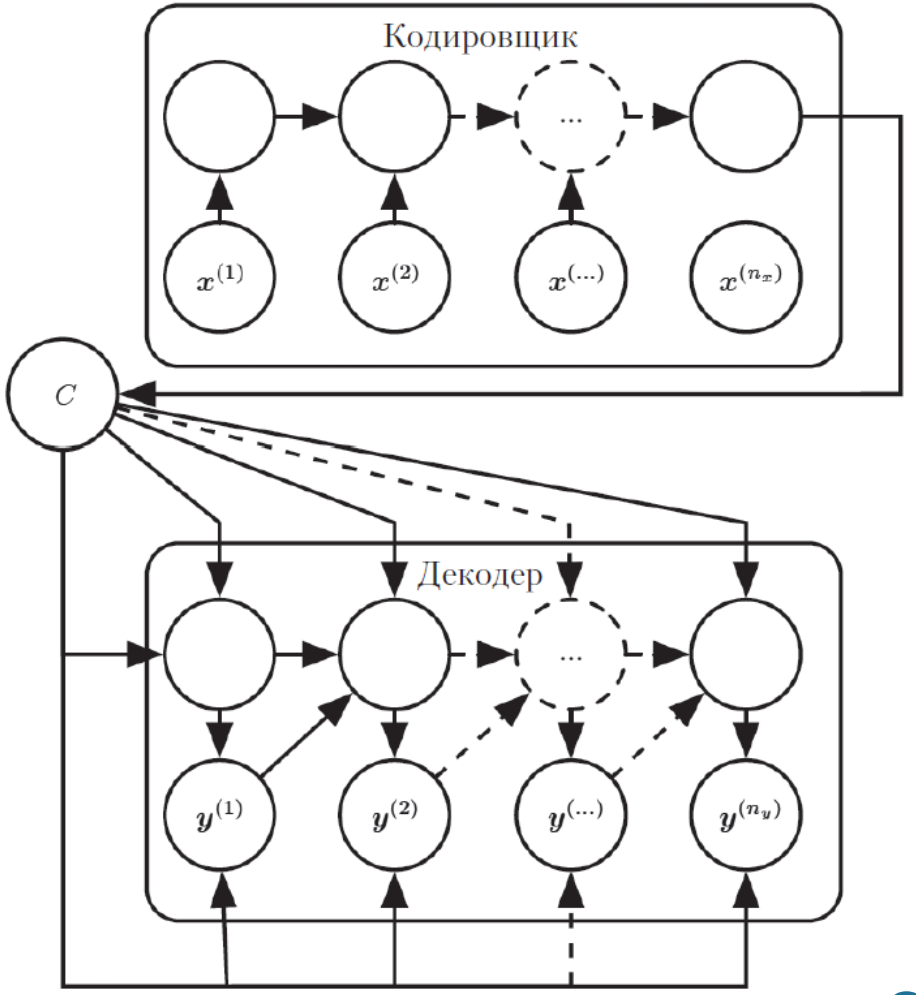
Вычисление типичной двунаправленной рекуррентной нейронной сети, которая должна обучиться отображать входные последовательности x на выходные последовательности y, с функцией потерь $L^{(t)}$ на каждом шаге t. Рекуррентные блоки h распространяют информацию вперед во времени (слева направо), а блоки g – назад во времени (справа налево). Таким образом, в каждой точке t выходные блоки $o^{(t)}$ могут пользоваться сводной информацией о прошлом из входа $h^{(t)}$ и сводной информацией о будущем из входа $g^{(t)}$



Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

АРХИТЕКТУРЫ КОДИРОВЩИК-ДЕКОДЕР ИЛИ ПОСЛЕДОВАТЕЛЬНОСТЬ В ПОСЛЕДОВАТЕЛЬНОСТЬ

Пример архитектуры РНС типа кодировщик-декодер, или последовательность в последовательность, которая обучается генерировать выходную последовательность $y^{(1)}$, …, $y^{(ny)}$ по входной последовательности $x^{(1)}$, $x^{(2)}$, …, $x^{(nx)}$. Сеть состоит из кодирующей РНС, которая читает входную последовательность, и декодирующей РНС, которая генерирует выходную последовательность (или вычисляет вероятность заданной выходной последовательности). Конечное скрытое состояние кодирующей РНС используется для вычисления контекстной переменной C фиксированного размера, которая представляет собой семантическую сводку входной последовательности и подается на вход декодирующей РНС.



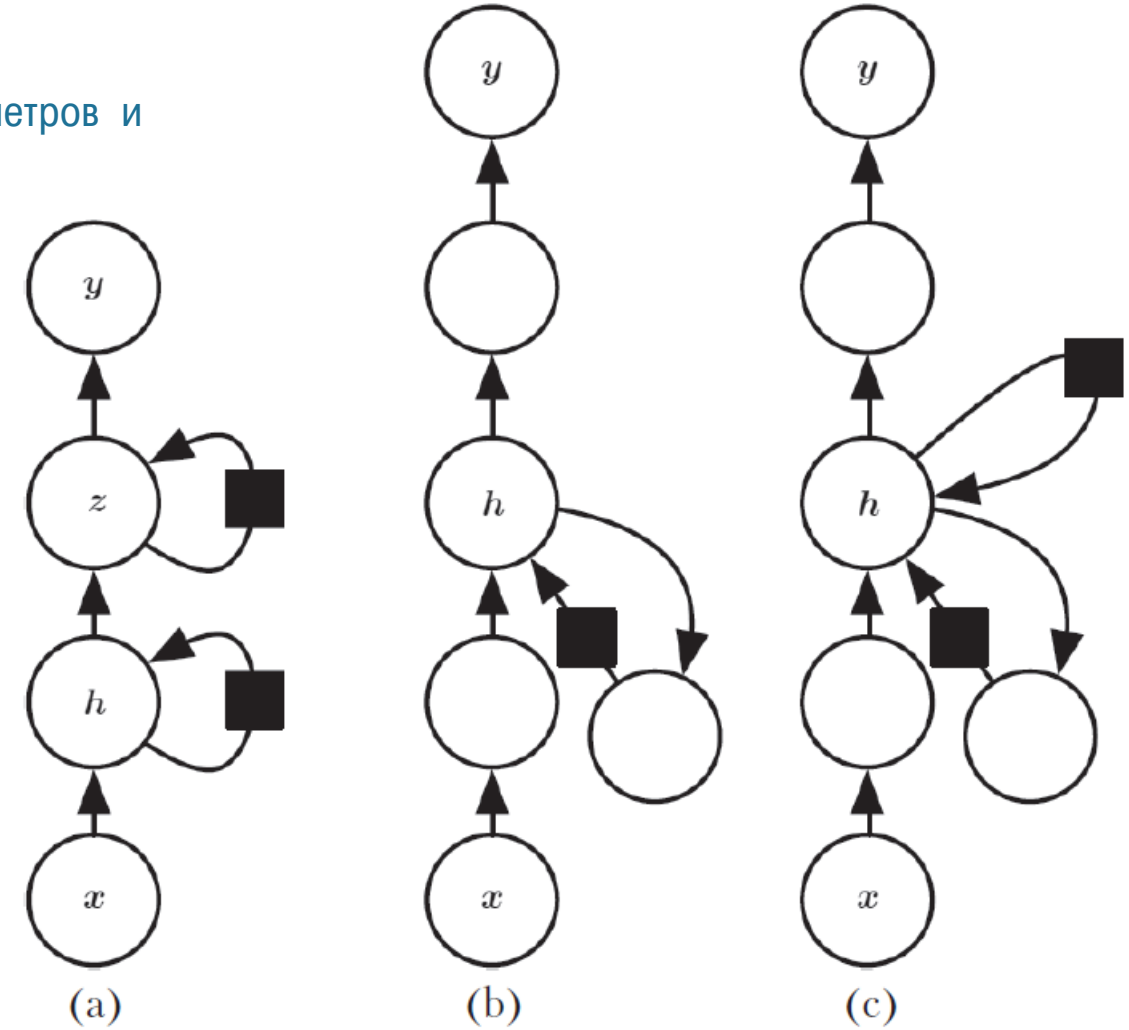
Лекция 2.5 – Моделирование рекуррентных и рекурсивных сетей

ГЛУБОКИЕ РЕКУРРЕНТНЫЕ СЕТИ

Вычисления в большинстве РНС можно разложить на три блока параметров и ассоциированные с ними преобразования:

- из входа в скрытое состояние;
- из предыдущего скрытого состояния в следующее;
- из скрытого состояния в выход.

Рекуррентную нейронную сеть можно сделать глубокой разными способами . (a) Скрытое рекуррентное состояние можно разделить на иерархически организованные группы. (b) Между входом и скрытым состоянием, между двумя скрытыми уровнями скрытого состояния и между скрытым состоянием и выходом можно поместить более глубокое вычисление (например, МСП). Это может удлинить кратчайший путь, соединяющий разные временные шаги. (c) Эффект удлинения пути можно сгладить путем добавления прямых связей

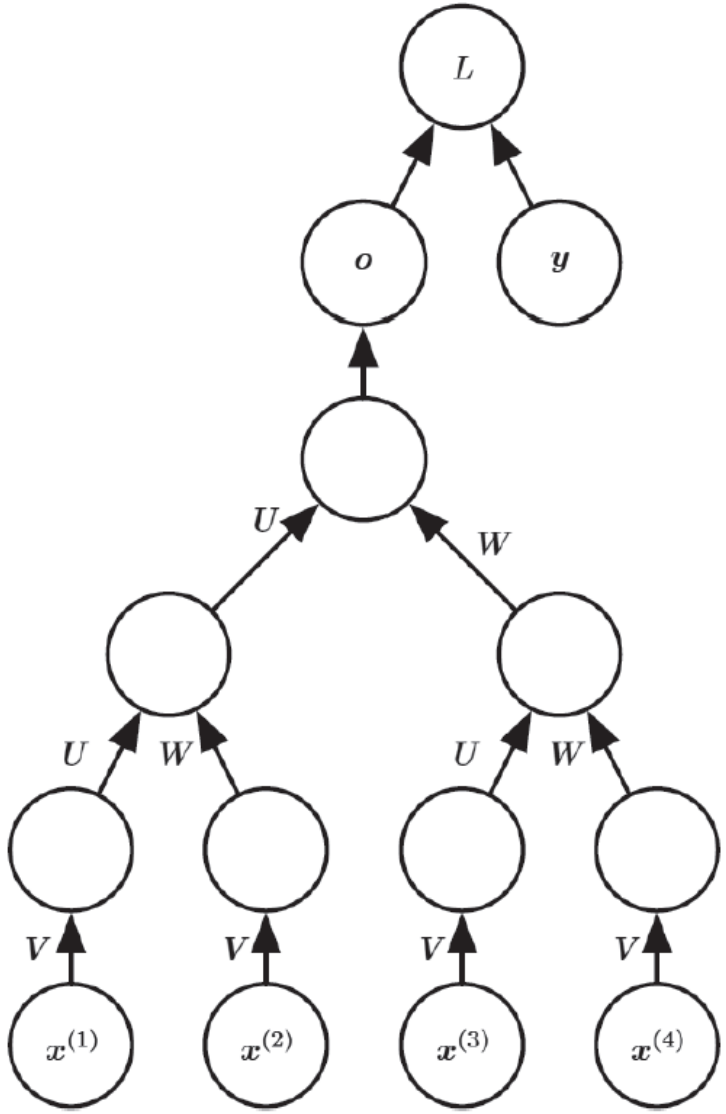


Лекция 2.5 – Моделирование рекуррентных и рекурсивных сетей

РЕКУРСИВНЫЕ НЕЙРОННЫЕ СЕТИ

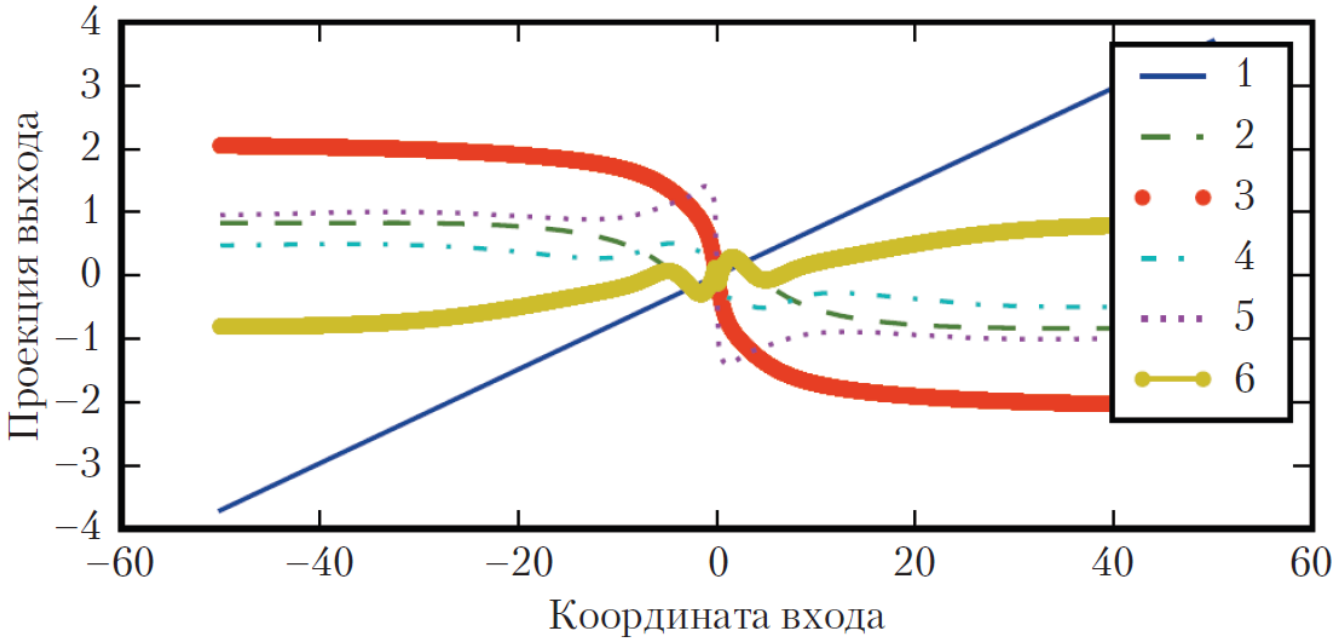
Рекурсивные нейронные сети – еще один вид обобщения рекуррентных сетей, для которого характерен граф вычислений, структурированный как глубокое дерево, а не как цепная структура, присущая РНС.

Граф вычислений рекурсивной сети является деревом, а не цепочкой, как в рекуррентных сетях. Последовательность переменной длины $x^{(1)}$, $x^{(2)}$, …, $x^{(t)}$ можно отобразить на представление фиксированного размера (выход o) с фиксированным набором параметров (матрицы весов U, V, W). На рисунке показан случай обучения с учителем, когда предъявляется метка y, ассоциированная со всей последовательностью.



Лекция 2.5 - Моделирование рекуррентных и рекурсивных сетей

ПРОБЛЕМА ДОЛГОСРОЧНЫХ ЗАВИСИМОСТЕЙ

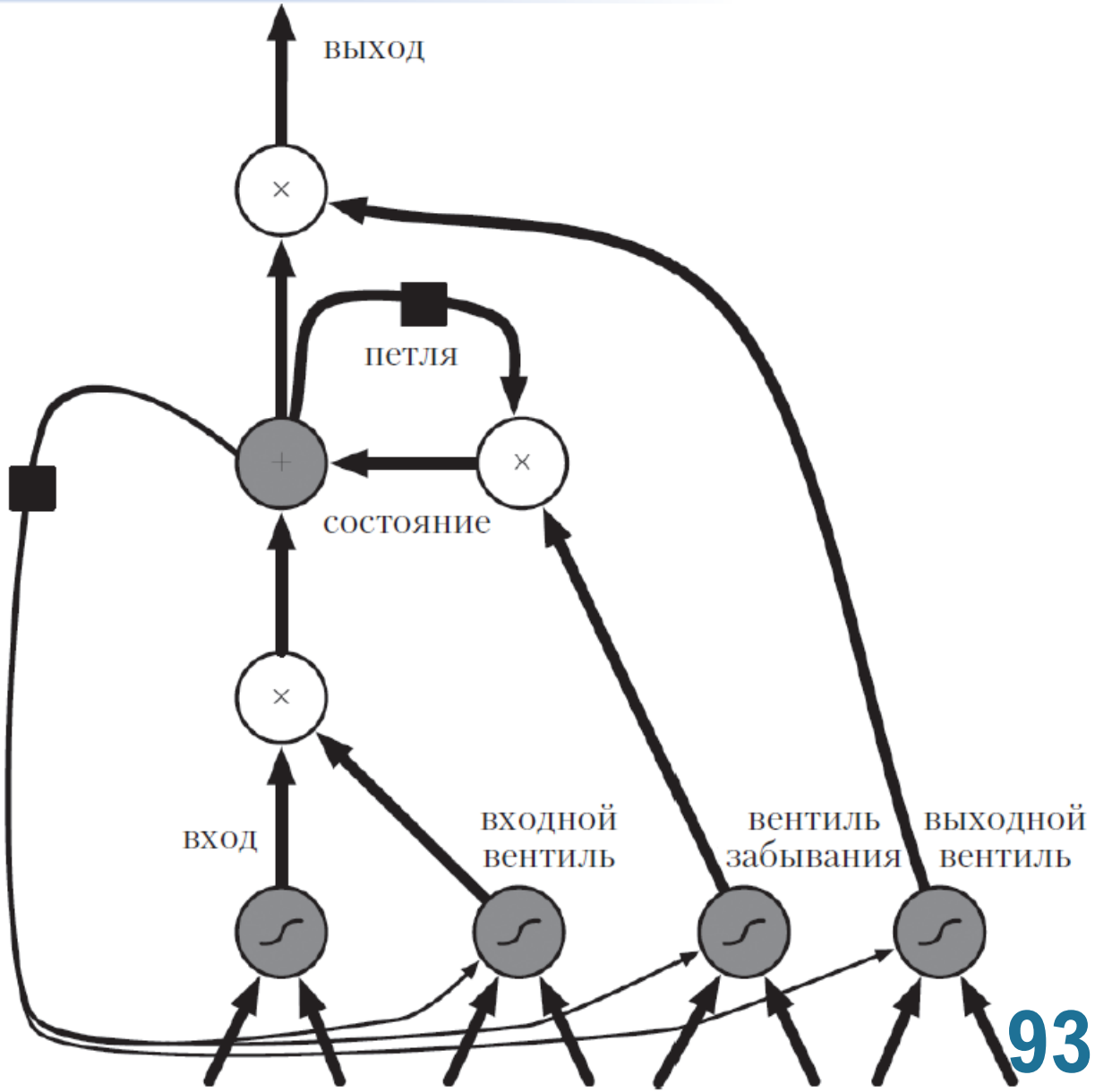


Повторная композиция функций. Многократная композиция нелинейной функции (например, показанного здесь гиперболического тангенса) приводит к сильно нелинейному результату; обычно в большинстве точек производная очень мала, в некоторых велика, и часто наблюдается переход от возрастания к убыванию и наоборот. На этом рисунке показана линейная проекция 100-мерного скрытого состояния на одно измерение, отложенное по оси y. По оси x отложена координата начального состояния вдоль случайно выбранного направления в 100-мерном пространстве. Таким образом, этот график можно рассматривать как сечение графика многомерной функции. На графиках показана функция после каждого временного шага, или, эквивалентно, результат многократной композиции функции перехода с самой собой

Лекция 2.5 – Моделирование рекуррентных и рекурсивных сетей

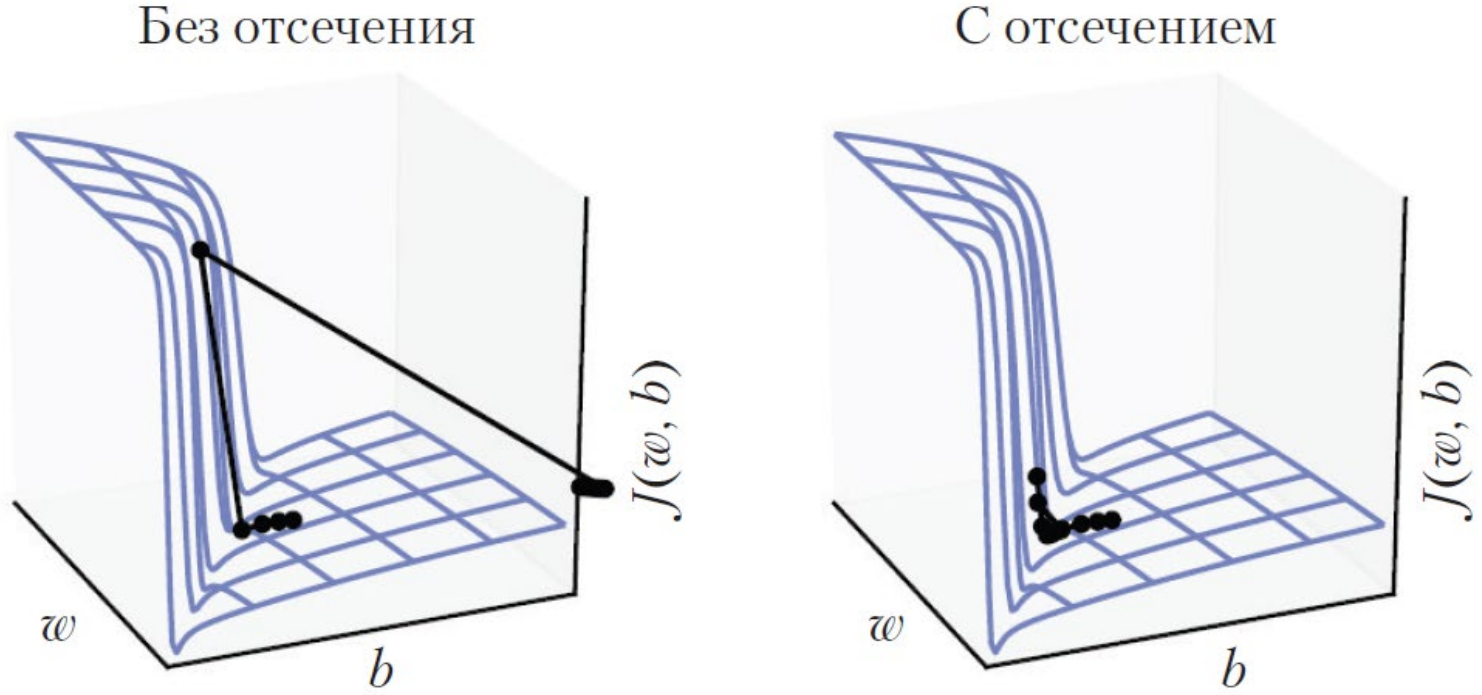
ДОЛГАЯ КРАТКОСРОЧНАЯ ПАМЯТЬ

Принципиальная схема «ячейки» рекуррентной LSTM-сети. Ячейки, рекуррентно связанные между собой, заменяют стандартные скрытые блоки в обыкновенных рекуррентных сетях. Входной признак вычисляется регулярным блоком искусственного нейрона. Его значение можно аккумулировать в состоянии, если сигмоидный входной вентиль это допускает. В блоке состояния имеется линейная петля, вес которой управляется вентилем забывания. Выход ячейки можно перекрыть с помощью выходного вентиля. Во всех вентильных блоках имеется сигмоидная нелинейность, тогда как во входном блоке разрешены произвольные сплющивающие нелинейности. Черным квадратиком обозначена задержка на одном временном шаге.



Лекция 2.5 – Моделирование рекуррентных и рекурсивных сетей

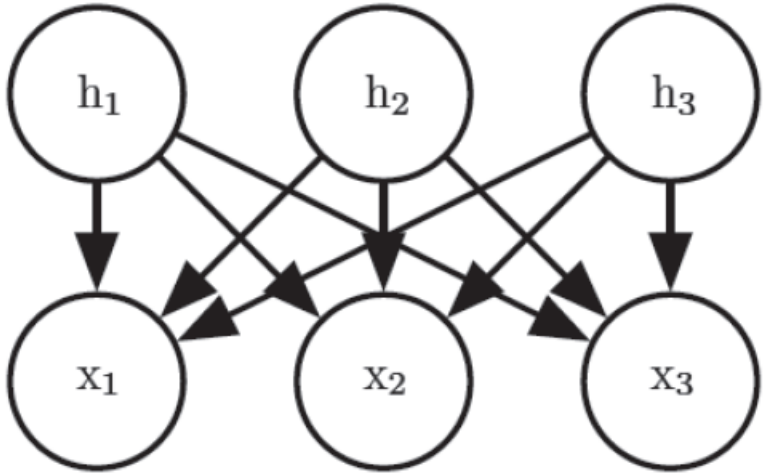
ОПТИМИЗАЦИЯ В КОНТЕКСТЕ ДОЛГОСРОЧНЫХ ЗАВИСИМОСТЕЙ



Пример эффекта отсечения градиента в рекуррентной сети с двумя параметрами w и b. В результате отсечения метод градиентного спуска иногда ведет себя более разумно вблизи особенно крутых уступов. Такие крутые уступы часто встречаются в рекуррентных сетях рядом с участками, где сеть ведет себя почти линейно. Крутизна уступа экспоненциально возрастает с увеличением числа временных шагов, поскольку на каждом шаге матрица весов умножается на себя же. (Слева) Градиентный спуск без отсечения градиента проскакивает дно этого небольшого ущелья, после чего градиент резко возрастает на стенке уступа. Большой градиент приводит к катастрофическому уходу параметров от оптимума. (Справа) Градиентный спуск с отсечением градиента реагирует на уступ не так резко. Хотя подъем имеет место, размер шага ограничен, поэтому мы не можем уйти слишком далеко от крутой области рядом с решением.

Лекция 2.6 - Линейные факторные модели

На передовых рубежах глубокого обучения часто требуется построить вероятностную модель входа, $p_{model}(x)$. В принципе, в такой модели может использоваться вероятностный вывод для предсказания любой переменной в ее окружении при известных других переменных. Во многих моделях имеются также латентные переменные h, так что $p_{model}(x) = \mathbb{E}h\ p_{model}(x \mid h)$.



Ориентированная графическая модель, описывающая семейство линейных факторных моделей. Предполагается, что наблюдаемый вектор x является линейной комбинацией независимых латентных факторов h и шума. В различных моделях, например в вероятностном методе главных компонент, факторном анализе и анализе независимых компонент (ICA), делаются различные предположения о форме шума и априорном распределении p(h)

Лекция 2.6 - Линейные факторные модели

ВЕРОЯТНОСТНЫЙ PCA И ФАКТОРНЫЙ АНАЛИЗ

Чтобы переформулировать метод главных компонент (PCA) в вероятностном контексте, мы можем внести небольшую модификацию в модель факторного анализа, сделав условные дисперсии σ_i^2 равными. В таком случае ковариация x равна просто $WW^\top + \sigma^2 I$, где σ^2 – теперь скаляр. Тогда условное распределение имеет вид:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{b}, \boldsymbol{W}\boldsymbol{W}^\top + \sigma^2 \boldsymbol{I})$$

АНАЛИЗ НЕЗАВИСИМЫХ КОМПОНЕНТ (ICA)

Анализ независимых компонент (independent component analysis – ICA) – один из самых старых алгоритмов обучения представлений Идея этого подхода к моделированию линейных факторов заключается в том, чтобы представить наблюдаемый сигнал в виде линейной комбинации нескольких составляющих. Предполагается, что эти сигналы полностью независимы, а не просто не коррелированы.

Лекция 2.6 - Линейные факторные модели

АНАЛИЗ МЕДЛЕННЫХ ПРИЗНАКОВ

Анализ медленных признаков (slow feature analysis – SFA) – линейная факторная модель, в которой для обучения инвариантных признаков используется информация от сигналов времени.

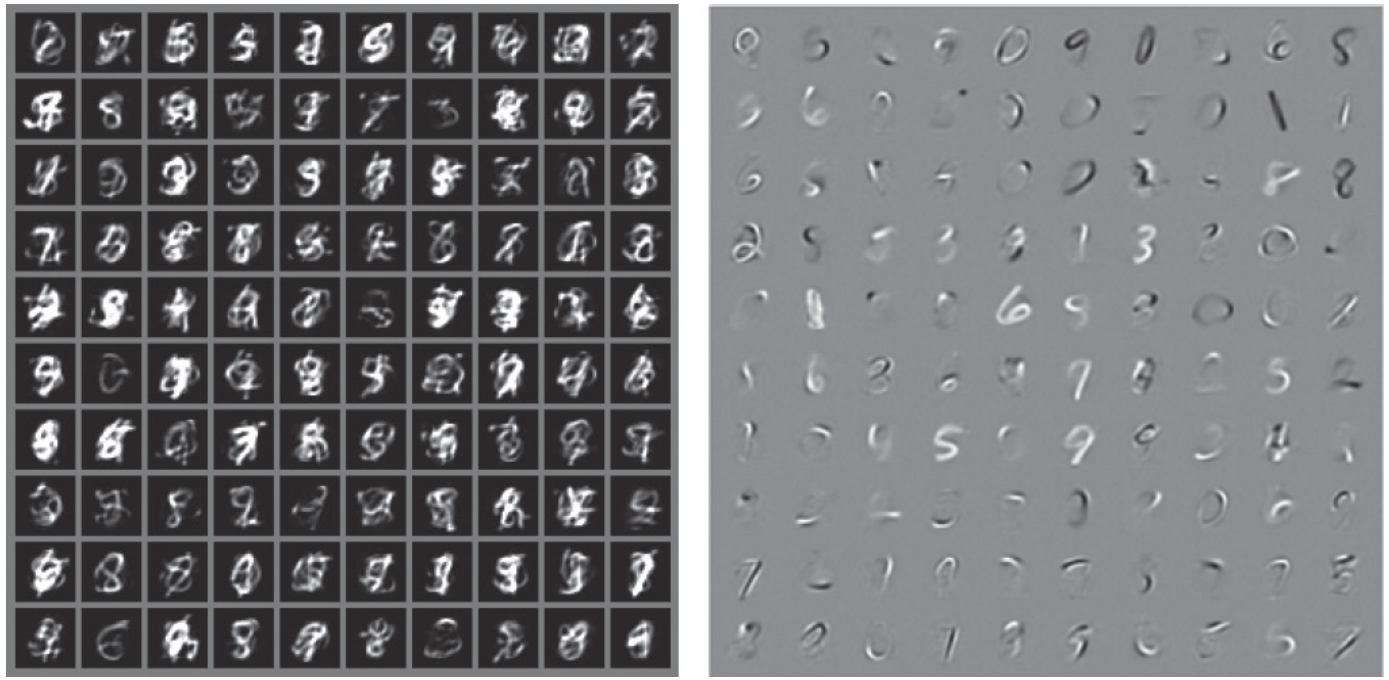
Главное достоинство SFA состоит в том, что можно теоретически предсказать, каким именно признакам обучится SFA, даже в глубокой нелинейной конфигурации. Для этого необходимо знать динамику окружения в терминах конфигурационного пространства (например, для случайного движения в трехмерном отрисованном окружении теоретический анализ основывается на знании распределения вероятности положения и скорости перемещения камеры). Зная, как на самом деле изменяются объясняющие факторы, можно аналитически найти оптимальные функции, выражающие эти факторы. На практике эксперименты с применением глубокой модели SFA к имитированным данным, похоже, восстанавливают теоретически предсказанные функции. Это большое преимущество, по сравнению с другими алгоритмами обучения, где функция стоимости сильно зависит от конкретных значений пикселей, из-за чего установить, каким функциям обучится модель, гораздо труднее.

РАЗРЕЖЕННОЕ КОДИРОВАНИЕ

Разреженное кодирование – это линейная факторная модель, которая активно изучалась как механизм обучения признакам без учителя и выделения признаков. Строго говоря, термин «разреженное кодирование» относится к процессу вывода значения h в этой модели, а «разреженное моделирование» – к процессу проектирования и обучения модели, но часто под «разреженным кодированием» понимают то и другое.

Лекция 2.6 - Линейные факторные модели

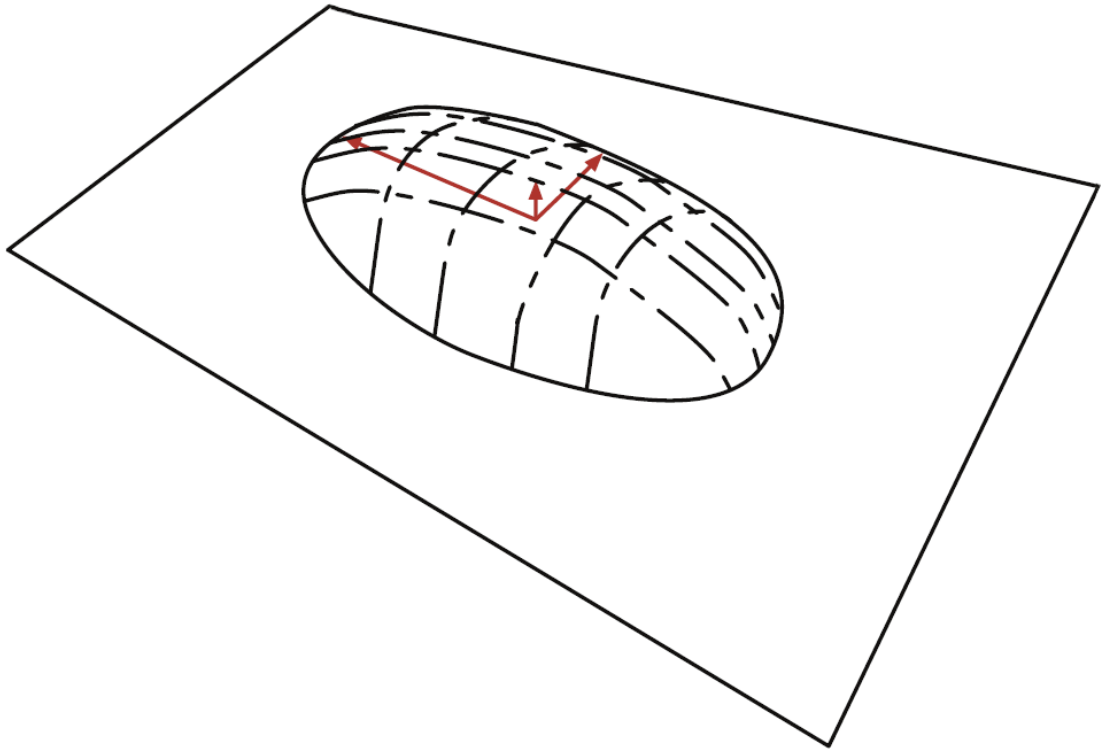
ИНТЕРПРЕТАЦИЯ PCA В ТЕРМИНАХ МНОГООБРАЗИЙ



Образцы примеров и весов из модели разреженного кодирования типа Spike-and-Slab, обученной на наборе данных MNIST. (Слева) Выборка из модели не похожа на обучающие примеры. На первый взгляд можно предположить, что модель плохо обучена. (Справа). Векторы весов модели обучены представлять росчерки пера и иногда целые цифры. Следовательно, модель обучилась полезным признакам. Проблема в том, что факторное априорное распределение признаков приводит к комбинированию случайных подмножеств признаков. Немногие из таких подмножеств годятся для образования распознаваемой цифры из набора MNIST. Отсюда необходимость разработки порождающих моделей с более мощными распределениями латентных кодов.

Лекция 2.6 - Линейные факторные модели

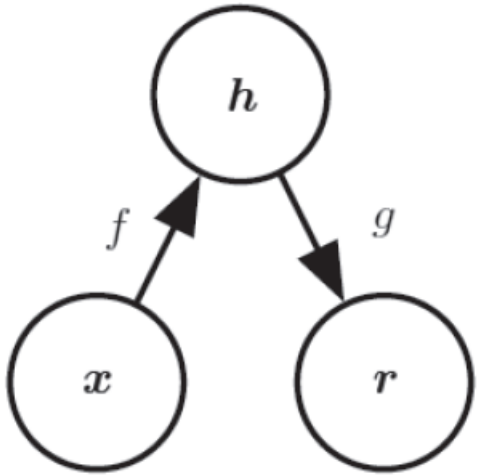
ИНТЕРПРЕТАЦИЯ PCA В ТЕРМИНАХ МНОГООБРАЗИЙ



Плоское нормальное распределение с высокой концентрацией вероятности в окрестности многообразия низкой размерности. На рисунке показана верхняя половина «блина» над «плоскостью многообразия», проходящей через его середину. Дисперсия в направлении, ортогональном многообразию, очень мала (стрелка в направлении наружу от плоскости) и может считаться «шумом», тогда как дисперсии в других направлениях (стрелки внутри плоскости) велики и соответствуют «сигналу» и определяют систему координат для данных пониженной размерности

Лекция 2.7 - 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).



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)

Лекция 2.7 - Автокодировщики

ПОНИЖАЮЩИЕ АВТОКОДИРОВЩИКИ

Процесс обучения описывается просто как минимизация функции потерь: $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$

Автокодировщики с нелинейными функциями кодирования f и декодирования g могут обучиться более мощным нелинейным обобщениям PCA.

РАЗРЕЖЕННЫЕ АВТОКОДИРОВЩИКИ

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

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

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.

Лекция 2.7 - Автокодировщики

ШУМОПОДАВЛЯЮЩИЕ АВТОКОДИРОВЩИКИ

Шумоподавляющий автокодировщик (denoising autoencoder – DAE) минимизирует функцию: $L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$

РЕГУЛЯРИЗАЦИЯ ПОСРЕДСТВОМ ШТРАФОВАНИЯ ПРОИЗВОДНЫХ

Еще одна стратегия регуляризации автокодировщика подразумевает использование штрафа, как в разреженных автокодировщиках, но с Ω другого вида:

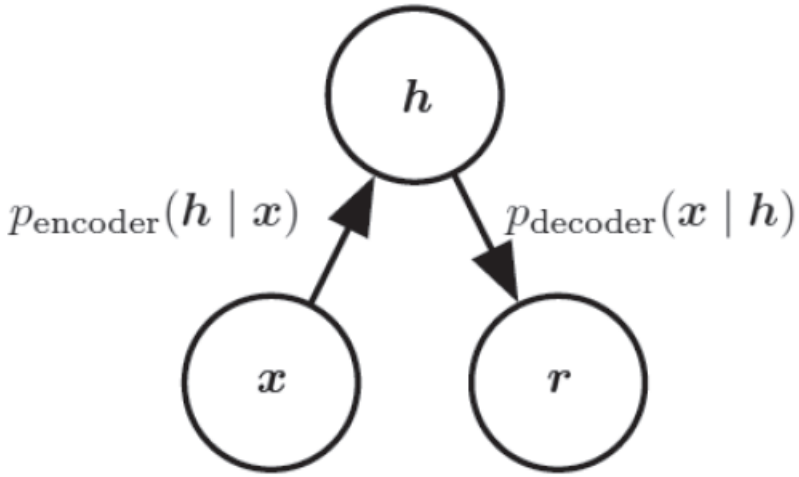
$$\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \sum_i \|\nabla_{\boldsymbol{x}} h_i\|^2$$

РЕПРЕЗЕНТАТИВНАЯ СПОСОБНОСТЬ, РАЗМЕР СЛОЯ И ГЛУБИНА

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.

Лекция 2.7 - Автокодировщики

СТОХАСТИЧЕСКИЕ КОДИРОВЩИКИ И ДЕКОДЕРЫ

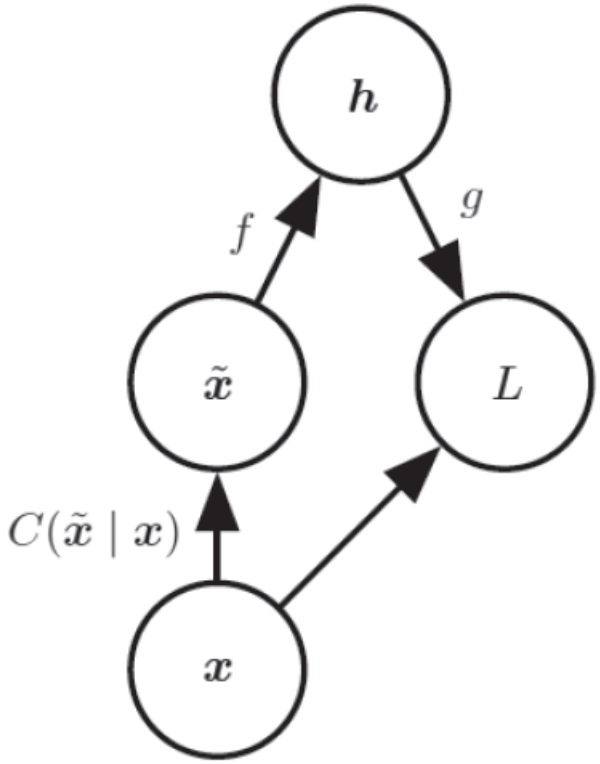


Структура стохастического автокодировщика, в которой кодировщик и декодер – функции, содержащие шум, т. е. их выход можно рассматривать как выборку из распределения $p_{encoder}(h \mid x)$ для кодировщика и $p_{decoder}(x \mid h)$ для декодера. В общем случае распределения кодировщика и декодера не обязательно являются условными распределениями, совместимыми с совместным распределением $p_{model}(x, h)$. Обучение кодировщика и декодера как шумоподавляющего автокодировщика делает их асимптотически совместимыми (при достаточной емкости и количестве обучающих примеров).

Лекция 2.7 - Автокодировщики

ШУМОПОДАВЛЯЮЩИЕ АВТОКОДИРОВЩИКИ

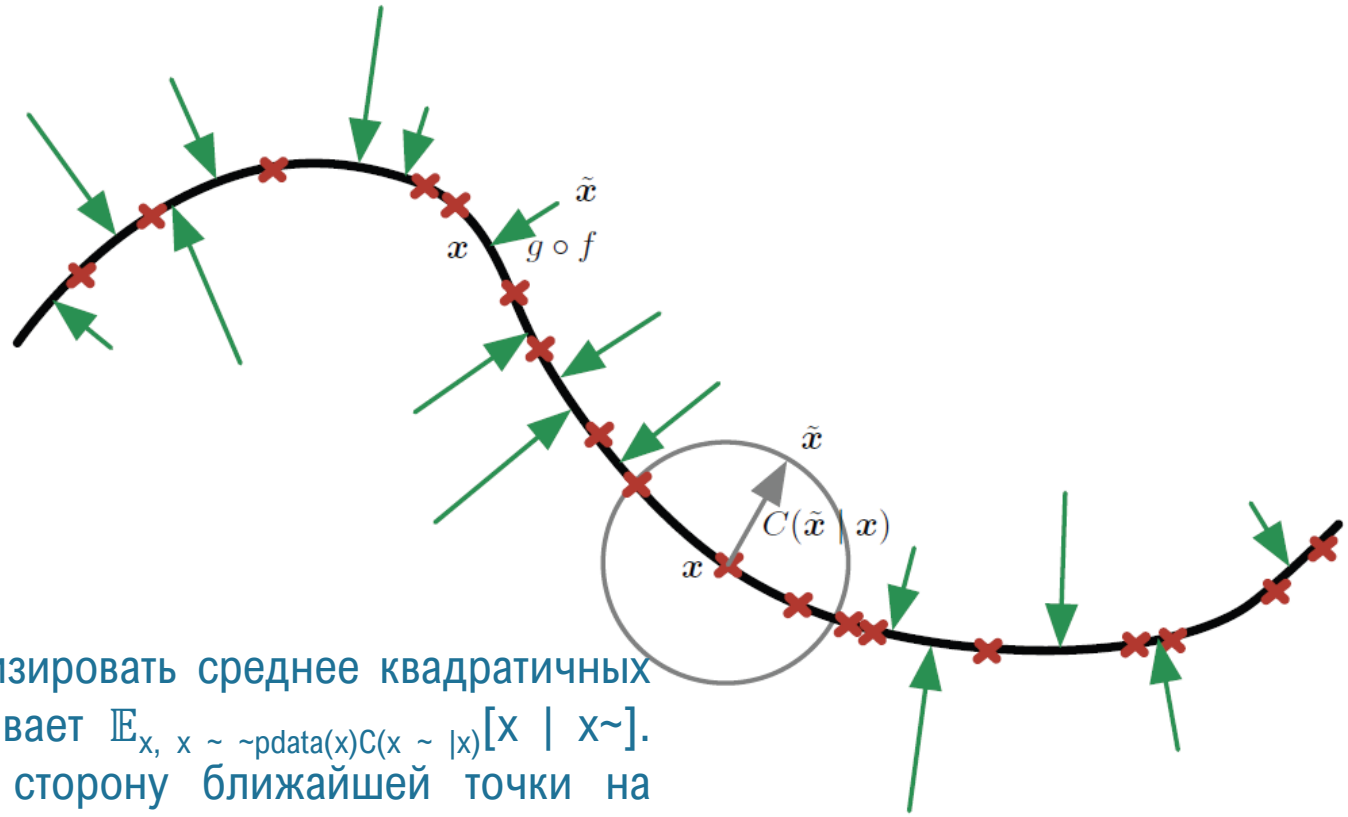
Шумоподавляющим автокодировщиком (DAE) называется автокодировщик, который получает на входе искаженные данные и обучается предсказывать истинные, неискаженные данные.



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

Лекция 2.7 - Автокодировщики

ШУМОПОДАВЛЯЮЩИЕ АВТОКОДИРОВЩИКИ

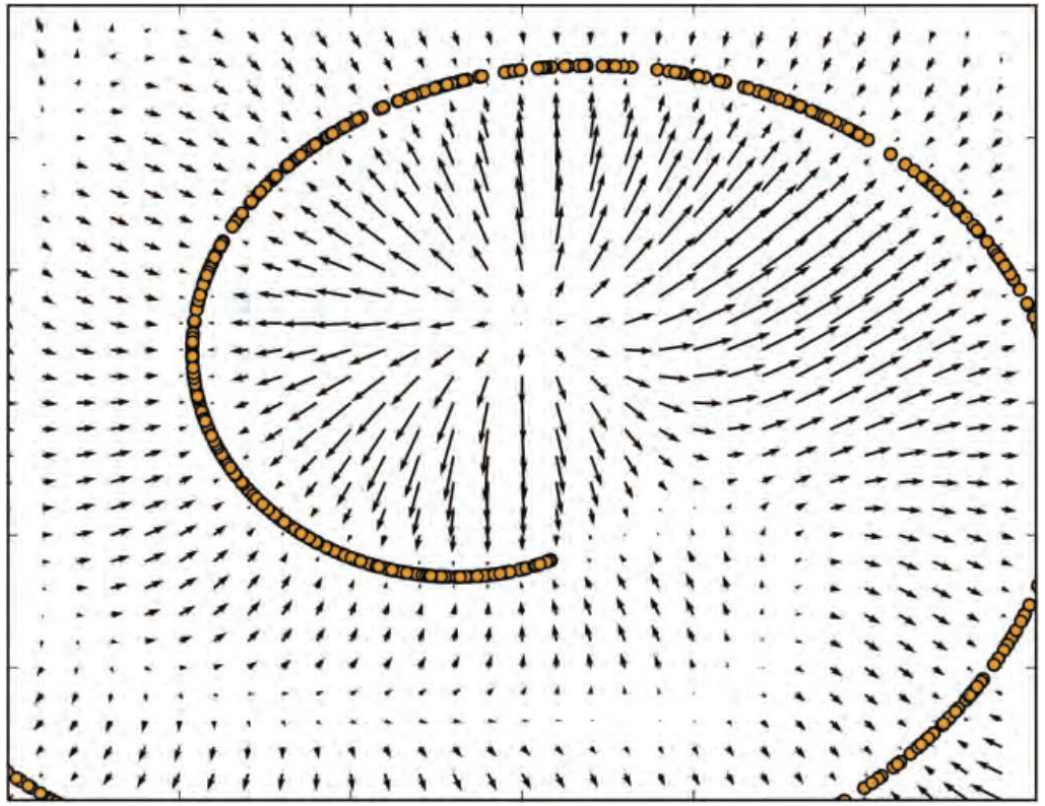


Шумоподавляющий автокодировщик обучается отображать искаженные данные x~ на исходные x. Обучающие примеры x обозначены красными крестиками, лежащими в окрестности многообразия низкой размерности, показанного жирной черной линией. Искажающий процесс C(x~ | x) представлен серой окружностью с равновероятными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

Лекция 2.7 - Автокодировщики

ШУМОПОДАВЛЯЮЩИЕ АВТОКОДИРОВЩИКИ

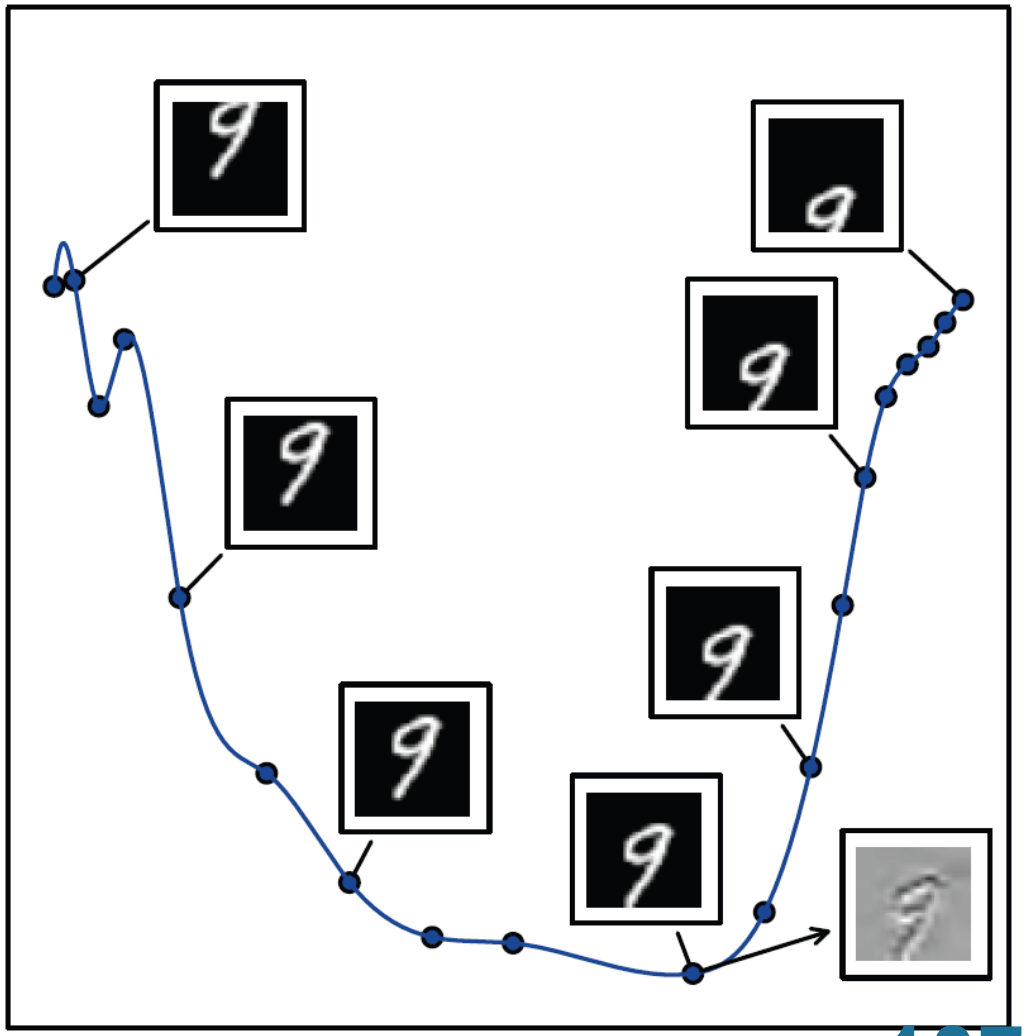
Обученное шумоподавляющим автокодировщиком векторное поле вокруг одномерного искривленного многообразия, в окрестности которого сконцентрированы двумерные данные. Каждая стрелка по длине пропорциональна реконструкции минус входной вектор автокодировщика и направлена в сторону увеличения вероятности в 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.



Лекция 2.7 - Автокодировщики

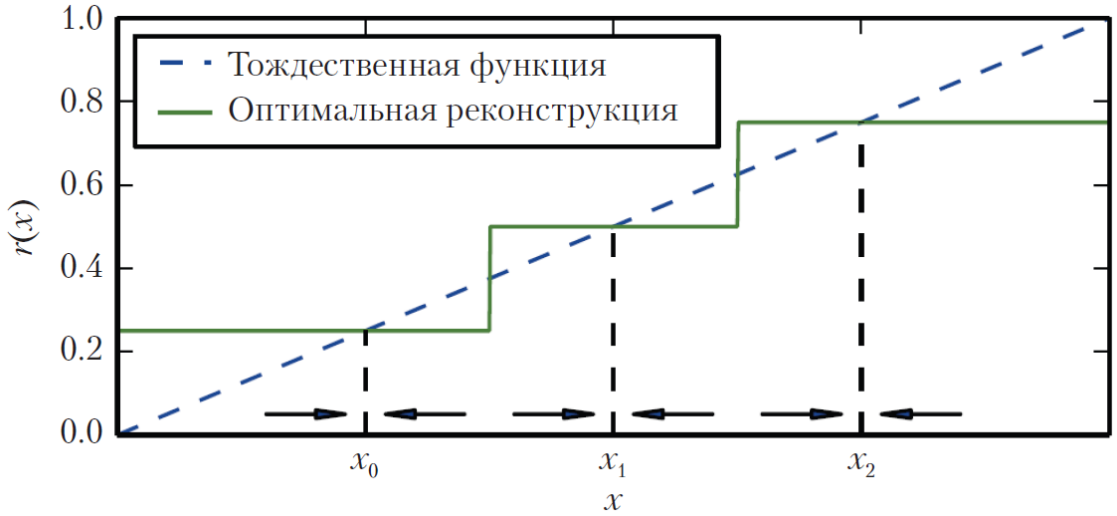
ОБУЧЕНИЕ МНОГООБРАЗИЙ С ПОМОЩЬЮ АВТОКОДИРОВЩИКОВ

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. 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



Лекция 2.7 - Автокодировщики

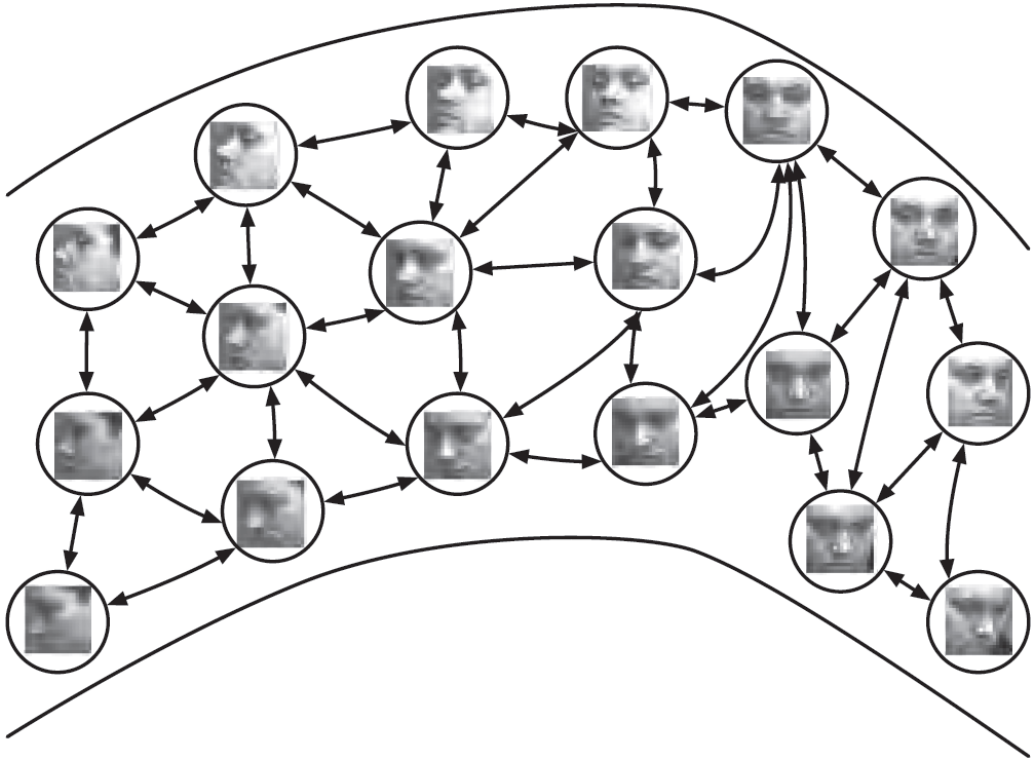
ОБУЧЕНИЕ МНОГООБРАЗИЙ С ПОМОЩЬЮ АВТОКОДИРОВЩИКОВ



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

Лекция 2.7 - Автокодировщики

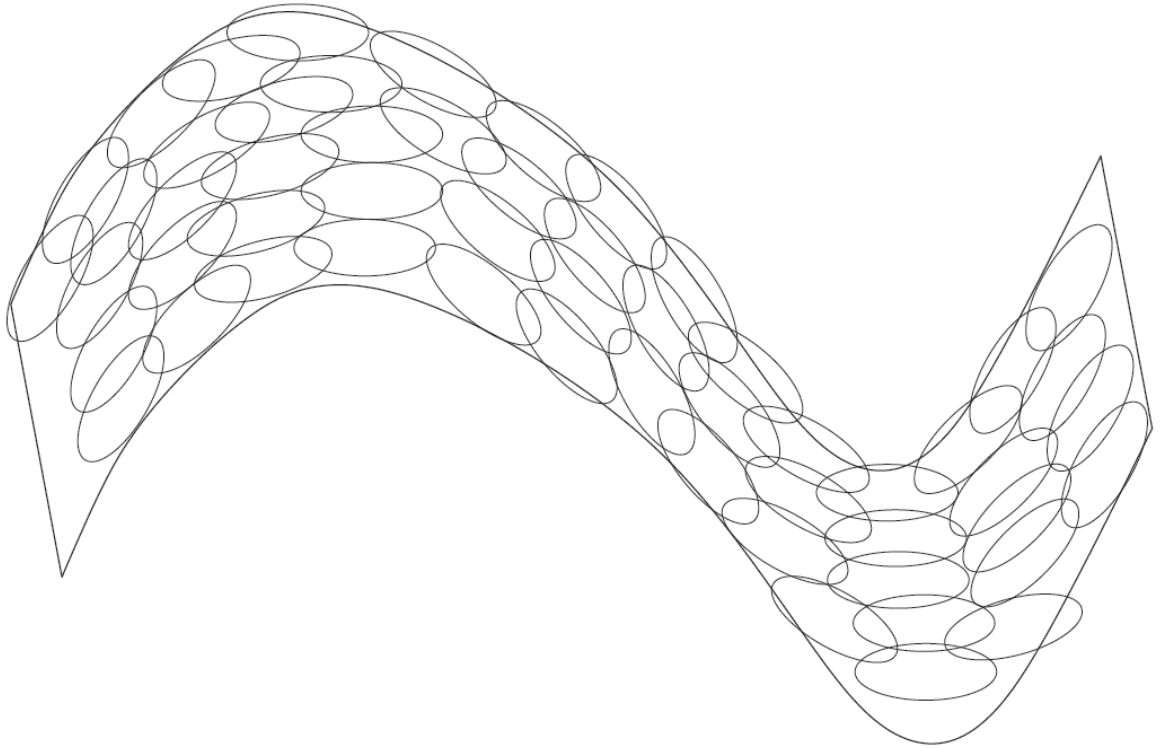
СЖИМАЮЩИЕ АВТОКОДИРОВЩИКИ



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.

Лекция 2.7 - Автокодировщики

СЖИМАЮЩИЕ АВТОКОДИРОВЩИКИ



Если касательные плоскости 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, как в методе окна Парзена для многообразий, или его нелокальном варианте, основанном на нейронной сети.

Лекция 2.7 - Автокодировщики

СЖИМАЮЩИЕ АВТОКОДИРОВЩИКИ

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, и CAE 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).

РАЗДЕЛ 3
ПРАКТИЧЕСКИЕ ПРИЛОЖЕНИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

Лекция 3.1 - Крупномасштабное глубокое обучение на многоядерных процессорах

КРУПНОМАСШТАБНОЕ ГЛУБОКОЕ ОБУЧЕНИЕ

Глубокое обучение основано на философии коннекционизма. Один из ключевых факторов, способствовавших повышению верности нейронных сетей и сложности решаемых ими задач в период начиная с 1980-х годов и по сегодняшний день, – кардинальное увеличение размера сетей. Поскольку размер нейронной сети играет решающую роль, для глубокого обучения необходимы высокопроизводительное оборудование и соответствующая программная инфраструктура

РЕАЛИЗАЦИИ НА БЫСТРЫХ CPU

Традиционно обучение нейронных сетей производилось на одной машине с одним центральным процессором (CPU). Сегодня это считается недостаточным. Мы используем в основном вычисления на графических процессорах (GPU) или на CPU многих машин, связанных в единую сеть. Но прежде чем перейти на такие дорогостоящие конфигурации, исследователям пришлось немало потрудиться, чтобы доказать, что CPU уже не справляются с вычислительной нагрузкой, необходимой нейронным сетям.

Лекция 3.1 - Крупномасштабное глубокое обучение на многоядерных процессорах

КРУПНОМАСШТАБНЫЕ РАСПРЕДЕЛЕННЫЕ РЕАЛИЗАЦИИ

Часто вычислительных ресурсов одной машины недостаточно. Поэтому хотелось бы распределить рабочую нагрузку обучения и вывода между несколькими машинами. Распределить вывод просто, потому что каждый входной пример можно обработать на отдельной машине. Это называется ***распараллеливанием по данным***. Существует также режим ***распараллеливания модели***, когда несколько машин совместно работает над одним примером, причем каждая машина выполняет свою часть модели. Это возможно как для обучения, так и для вывода. Распараллеливание по данным на этапе обучения несколько сложнее. Мы можем увеличить размер мини-пакета, используемого на одном шаге СГС, но обычно выигрыш в терминах производительности оптимизации получается меньше линейного.

Было бы лучше, чтобы несколько машин параллельно вычисляли несколько шагов градиентного спуска. К сожалению, в стандартной формулировке алгоритм градиентного спуска строго последовательный: градиент на шаге t является функцией параметров, найденных на шаге t – 1. Эту проблему можно решить с помощью асинхронного стохастического градиентного спуска. В этом случае несколько процессорных ядер сообща использует память для представления параметров. Каждое ядро читает параметры без блокировки, затем вычисляет градиент, после чего инкрементирует параметры без блокировки. Это уменьшает среднее улучшение на каждом шаге градиентного спуска, потому что некоторые ядра перезаписывают достигнутое другими ядрами, но за счет увеличенного темпа выполнения шагов процесс обучения в целом протекает быстрее.

Лекция 3.1 - Крупномасштабное глубокое обучение на многоядерных процессорах

ДИНАМИЧЕСКАЯ СТРУКТУРА

Одна из общих стратегий ускорения систем обработки данных – построить систему с динамической структурой в виде графа, описывающего вычисления, необходимые для обработки входных данных. Системы обработки данных могут динамически определить, какую часть большого множества нейронных сетей выполнять для заданного входа. Отдельные нейронные сети также могут иметь внутреннюю динамическую структуру, т. е. определять, какое подмножество признаков (скрытых блоков) вычислять при имеющейся входной информации. Такую форму динамической структуры внутри нейронной сети иногда называют условным вычислением. Поскольку многие компоненты архитектуры могут иметь отношение только к небольшому количеству возможных входов, система будет работать быстрее, если эти признаки вычисляются только по мере необходимости.

СПЕЦИАЛИЗИРОВАННЫЕ АППАРАТНЫЕ РЕАЛИЗАЦИИ ГЛУБОКИХ СЕТЕЙ

Хотя в программных реализациях для процессоров общего назначения (CPU и GPU) обычно используются 32- или 64-разрядные числа с плавающей точкой, давно известно, что можно обойтись и меньшей точностью, по крайней мере на этапе вывода. В последние годы этот вопрос приобрел особую остроту, поскольку глубокое обучение стало широко использоваться в промышленных изделиях, а на примере GPU продемонстрирован серьезный выигрыш, который может дать более быстрое оборудование. Еще одним фактором, стимулирующим исследования в области специализированного оборудования для глубокого обучения, является замедление прогресса в разработке одного ядра CPU и GPU; теперь повышение быстродействия происходит в основном за счет распараллеливания обработки между несколькими ядрами (как в CPU, так и в GPU). Таким образом, специализированное оборудование – это способ выйти за привычные рамки во времена, когда проектируется новое оборудование для устройств с низким энергопотреблением.

РЕАЛИЗАЦИИ НА GPU

Большинство современных реализаций нейронных сетей основано на использовании графических процессоров (GPU) – специализированного оборудования, которое изначально разрабатывалось для графических приложений. Алгоритмам нейронных сетей свойственны такие же характеристики производительности, как для графических алгоритмов реального времени. В нейронных сетях обычно имеется много больших буферов параметров, значений активации и градиентов, каждый из которых необходимо полностью обновлять на каждом шаге обучения. Эти буферы не помещаются в кэш стандартного настольного компьютера, поэтому пропускная способность памяти часто оказывается лимитирующим фактором. Важное преимущество GPU над CPU заключается именно в высокой пропускной способности памяти.

В алгоритмах обучения нейронных сетей обычно не так много ветвлений и сложных средств управления потоком вычислений, поэтому они пригодны для работы на GPU. Поскольку нейронную сеть можно разложить на множество отдельных «нейронов», обрабатываемых независимо от других нейронов в том же слое, то для их обучения средства распараллеливания, имеющиеся в GPU, оказываются весьма кстати.

Лекция 3.2 - Крупномасштабное глубокое обучение на графических процессорах

РЕАЛИЗАЦИИ НА GPU

Использование графических карт для обучения нейронных сетей испытало настоящий взрыв популярности с появлением GPU общего назначения (GP-GPU), которые могут исполнять произвольный код, а не только подпрограммы отрисовки. Похожий на C язык программирования CUDA, разработанный компанией NVIDIA, дал средства для написания произвольного кода. Благодаря сравнительно удобной модели программирования, массовому параллелизму и высокой пропускной способности памяти GP-GPU стали идеальной платформой для программирования нейронных сетей.

Код для GPU принципиально многопоточный, и потоки необходимо тщательно синхронизировать. Например, операции с памятью выполняются быстрее, если их можно ***объединить***.

Еще один важный аспект программы для GPU – следить за тем, чтобы все потоки в группе выполняли одну и ту же команду одновременно. Это означает, что реализовать ветвление на GPU трудно. Потоки разбиваются на небольшие группы, называемые канатами (warp). Каждый поток каната в каждом цикле выполняет одну и ту же команду, поэтому если два потока из одного каната должны пройти по разным путям в коде, то проходить их придется последовательно, а не параллельно.

Лекция 3.2 - Крупномасштабное глубокое обучение на графических процессорах

СЖАТИЕ МОДЕЛИ

Во многих случаях у конечного пользователя ресурсов меньше, чем у разработчика. Например, сеть распознавания речи можно обучить на кластере из мощных компьютеров, а затем установить в мобильный телефон. Ключевая стратегия сокращения стоимости вывода – ***сжатие модели***.

Основная идея – заменить исходную дорогостоящую модель другой, потребляющей меньше памяти и работающей быстрее. Сжатие модели применимо, когда размер исходной модели обусловлен в основном стремлением предотвратить переобучение. В большинстве случаев модель с наименьшей ошибкой обобщения представляет собой ансамбль нескольких независимо обученных моделей. Вычислять предсказание всех n членов ансамбля дорого. Иногда даже одна модель обобщается лучше, если она велика (например, если применялась регуляризация прореживанием).

Такие большие модели обучают некоторую функцию f(x), но используют при этом гораздо больше параметров, чем необходимо для решения задачи. Их размер велик только потому, что число обучающих примеров ограничено. После того как функция f(x) аппроксимирована, мы можем сгенерировать обучающий набор, содержащий бесконечно много примеров, просто применив f к случайной выборке x. Затем мы обучаем новую, меньшую модель, так чтобы она совпадала с f(x) на этой выборке. Чтобы емкость новой модели использовалась наиболее эффективно, лучше выбирать новые точки x из распределения, похожего на реальные тестовые данные, которые будут предъявлены модели впоследствии. Это можно сделать, слегка исказив обучающие примеры или произведя выборку из порождающей модели, обученной на исходном обучающем наборе.

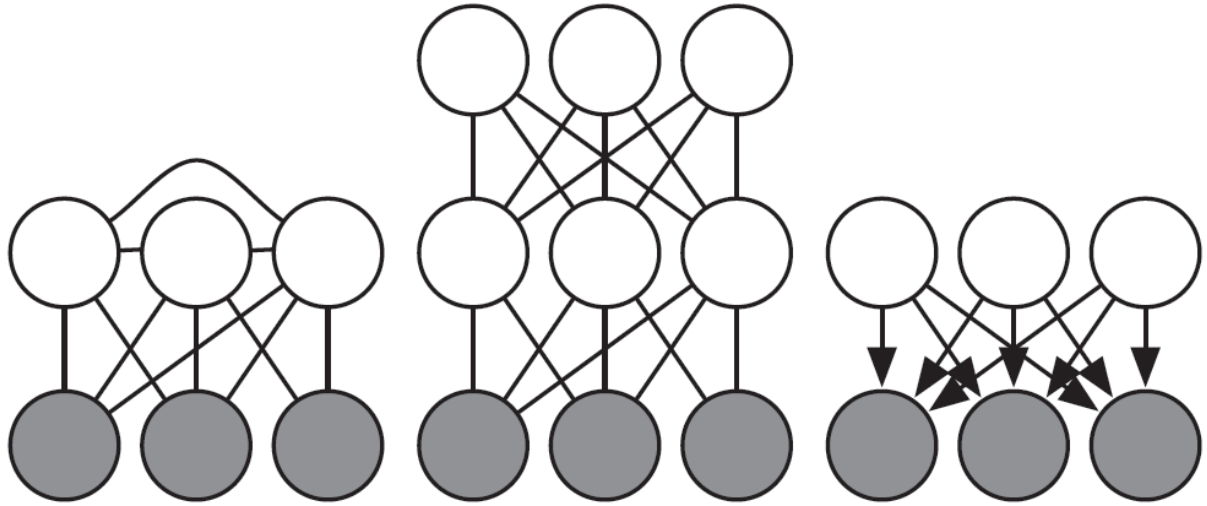
Многие вероятностные модели сложно обучить из-за трудностей вывода. В контексте глубокого обучения у нас обычно имеется множество наблюдаемых переменных v и множество латентных переменных h. Говоря о трудности вывода, мы обычно имеем в виду проблему вычисления p(h | v) или математических ожиданий относительно этого распределения.

ВЫВОД КАК ОПТИМИЗАЦИЯ

Во многих подходах к решению проблемы трудного вывода используется тот факт, что точный вывод можно описать как задачу оптимизации. Поэтому можно построить приближенный алгоритм вывода, аппроксимируя соответствующую задачу оптимизации. Для построения задачи оптимизации предположим, что имеется вероятностная модель, содержащая наблюдаемые переменные v и латентные переменные h. Мы хотели бы вычислить логарифмическое распределение вероятности наблюдаемых данных log p(v; θ). Иногда вычислить log p(v; θ) слишком трудно, если исключение h обходится дорого. Вместо этого мы можем вычислить нижнюю границу $\mathcal{L}(v, \theta, q)$ величины log p(v; θ). Эта граница называется ***нижней границей свидетельств*** (evidence lower bound – ELBO), а также ***отрицательной вариационной свободной энергией***.

Лекция 3.3 - Нечеткие модели и методы в глубоком обучении

ВЫВОД КАК ОПТИМИЗАЦИЯ



Неразрешимые проблемы вывода в глубоком обучении обычно являются результатом взаимодействий между латентными переменными в структурной графической модели. Эти взаимодействия могут быть вызваны ребрами, непосредственно соединяющими две латентные переменные, или более длинными путями, которые активируются, когда потомок V-структуры – наблюдаемая переменная. (Слева) Полуограниченная машина Больцмана со связями между скрытыми блоками. Из-за этих прямых связей между латентными переменными апостериорное распределение неразрешимо, поскольку имеются большие клики латентных переменных. (В центре) Для глубокой машины Больцмана,
состоящей из слоев переменных без внутрислойных связей, апостериорное распределение все равно неразрешимо из-за связей между слоями. (Справа) В этой ориентированной модели взаимодействия между латентными переменными присутствуют, когда видимые переменные наблюдаемые, поскольку каждые две латентные переменные являются родителями одной видимой. В некоторых вероятностных моделях вывод латентных переменных все-таки возможен, несмотря на изображенные на рисунке графические структуры. Так бывает, когда условные распределения вероятности выбраны так, что вносят дополнительные отношения независимости помимо представленных в графе. Например, в вероятностном методе главных компонент структура графа такая, как на правом рисунке, но вывод тем не менее простой благодаря специальным свойствам конкретных условных распределений (линейная комбинация нормальных распределений с взаимно ортогональными базисными векторами).

EM-АЛГОРИТМ

Первый основанный на максимизации нижней границы $\mathcal{L}$ алгоритм, который мы рассмотрим, – это популярный EM-алгоритм (expectation maximization) обучения моделей с латентными переменными.

EM-алгоритм состоит из двух чередующихся шагов, повторяемых до достижения сходимости:

- E-шаг (вычисление математического ожидания). Обозначим $\theta^{(0)}$ значение параметров в начале шага. Положим $q(h^{(i)} \mid v) = p(h^{(i)} \mid v^{(i)}; \theta^{(0)})$ для всех индексов i примеров $v^{(i)}$, на которых мы производим обучение (допустимы оба варианта: пакетный и мини-пакетный). Таким образом, q определено в терминах текущего значения параметра $\theta^{(0)}$; если мы изменим θ, то $p(h \mid v; \theta)$ изменится, но $q(h \mid v)$ останется равным $= p(h \mid v; \theta^{(0)})$;
- M-шаг (максимизация). Полностью или частично максимизировать выражение.

$$\sum_i \mathcal{L}(\boldsymbol{v}^{(i)}, \boldsymbol{\theta}, q)$$

EM-алгоритм содержит несколько разных идей. Прежде всего присутствует базовая структура процесса обучения, согласно которой мы обновляем параметры модели с целью повысить правдоподобие пополненного набора данных, в котором все отсутствующие переменные получили значения, предлагаемые оценкой апостериорного распределения.

MAP-ВЫВОД И РАЗРЕЖЕННОЕ КОДИРОВАНИЕ

Обычно термин вывод относится к вычислению распределения вероятности одного множества переменных при условии другого множества. При обучении вероятностных моделей с латентными переменными нас обычно интересует вычисление p(h | v). Альтернативная форма вывода заключается в вычислении одного самого вероятного значения отсутствующих переменных, а не вывода всего распределения их возможных значений. В контексте моделей с латентными переменными это означает, что нужно вычислить вывод апостериорного максимума:

$$\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v})$$

MAP-вывод обычно используется в глубоком обучении для выделения признаков и в качестве механизма обучения. Основное применение он находит в моделях разреженного кодирования.

ВАРИАЦИОННЫЙ ВЫВОД И ОБУЧЕНИЕ

Мы видели, что нижняя граница свидетельств $\mathcal{L}(v, \theta, q)$ является нижней границей $\log p(v; \theta)$, что вывод можно рассматривать как максимизацию $\mathcal{L}$ относительно q, а обучение – как максимизацию $\mathcal{L}$ относительно θ. Мы видели, что EM-алгоритм позволяет делать большие шаги обучения с фиксированным q и что алгоритмы обучения, основанные на MAP-выводе, дают возможность обучать модель с применением точечной оценки $p(h \mid v)$, не выводя всего распределения целиком. Теперь разработаем более общий подход к вариационному обучению.

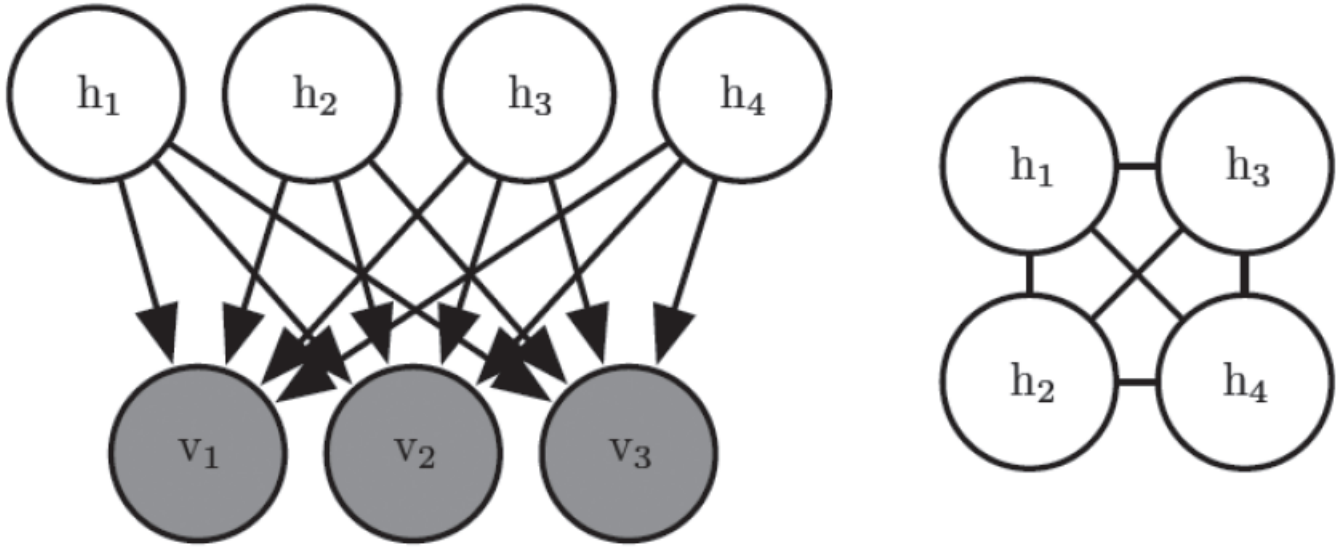
Основная идея вариационного обучения заключается в том, чтобы максимизировать $\mathcal{L}$ на ограниченном семействе распределений q. Это семейство следует выбирать так, чтобы было легко вычислить $\mathbb{E}_q \log p(h \mid v)$. Обычно для этого вводятся предположения о способе представления q в виде произведения.

Типичный подход к вариационному обучению – потребовать, чтобы q было факторным распределением:

$$q(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i q(h_i \mid \boldsymbol{v})$$

Это так называемый ***подход среднего поля***.

ДИСКРЕТНЫЕ ЛАТЕНТНЫЕ ПЕРЕМЕННЫЕ



Структура графа модели бинарного разреженного кодирования с четырьмя скрытыми блоками. (Слева) Структура графа p(h, v). Отметим, что ребра ориентированные и что два любых скрытых блока являются родителями каждого видимого блока. (Справа) Структура графа p(h | v). Чтобы учесть активные пути между сородителями, в апостериорном распределении должно быть проведено ребро между любыми двумя скрытыми блоками.

ДИСКРЕТНЫЕ ЛАТЕНТНЫЕ ПЕРЕМЕННЫЕ

Приступая к выводу уравнений вариационного обучения в модели бинарного разреженного кодирования, покажем, что благодаря использованию аппроксимации среднего поля обучение становится вычислительно разрешимым.

Нижняя граница свидетельств имеет вид:

$$
\begin{aligned}
&\mathcal{L}(\boldsymbol{v}; \boldsymbol{\theta}; q) \\
&= \mathbb{E}_{\mathbf{h}\sim q}[\log p(\boldsymbol{h}, \boldsymbol{v})] + H(q) \\
&= \mathbb{E}_{\mathbf{h}\sim q}[\log p(\boldsymbol{h}) + \log p(\boldsymbol{v} \mid \boldsymbol{h}) - \log q(\boldsymbol{h} \mid \boldsymbol{v})] \\
&= \mathbb{E}_{\mathbf{h}\sim q}\left[\sum_{i=1}^{m} \log p(h_i) + \sum_{i=1}^{n} \log p(v_i \mid \boldsymbol{h}) - \sum_{i=1}^{m} \log q(h_i \mid \boldsymbol{v})\right] \\
&= \sum_{i=1}^{m} [\hat{h}_i(\log\sigma(b_i) - \log\hat{h}_i) + (1 - \hat{h}_i)(\log\sigma(-b_i) - \log(1 - \hat{h}_i))] \\
&\quad + \mathbb{E}_{\mathbf{h}\sim q}\left[\sum_{i=1}^{m} \log\sqrt{\frac{\beta_i}{2\pi}} \exp\left(-\frac{\beta_i}{2}(v_i - \boldsymbol{W}_{i,:}\boldsymbol{h})^2\right)\right] \\
&= \sum_{i=1}^{m} [\hat{h}_i(\log\sigma(b_i) - \log\hat{h}_i) + (1 - \hat{h}_i)(\log\sigma(-b_i) - \log(1 - \hat{h}_i))] \\
&\quad + \frac{1}{2}\sum_{i=1}^{n}\left[\log\frac{\beta_i}{2\pi} - \beta_i\left(v_i^2 - 2v_i\boldsymbol{W}_{i,:}\hat{\boldsymbol{h}} + \sum_j\left[W_{i,j}^2\hat{h}_j + \sum_{k\neq j} W_{i,j}W_{i,k}\hat{h}_j\hat{h}_k\right]\right)\right].
\end{aligned}
$$

Лекция 3.4 - Глубокие порождающие модели

МАШИНЫ БОЛЬЦМАНА

$$P(\boldsymbol{x}) = \frac{\exp(-E(\boldsymbol{x}))}{Z}$$

$$E(\boldsymbol{x}) = -\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}$$

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\top \boldsymbol{R}\boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{h}^\top \boldsymbol{S}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}$$

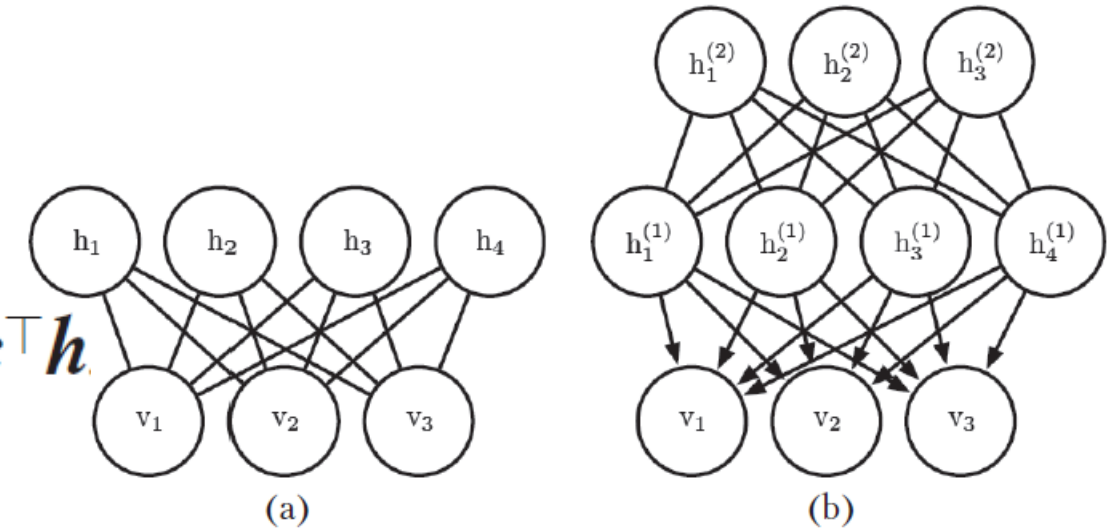
ОГРАНИЧЕННЫЕ МАШИНЫ БОЛЬЦМАНА

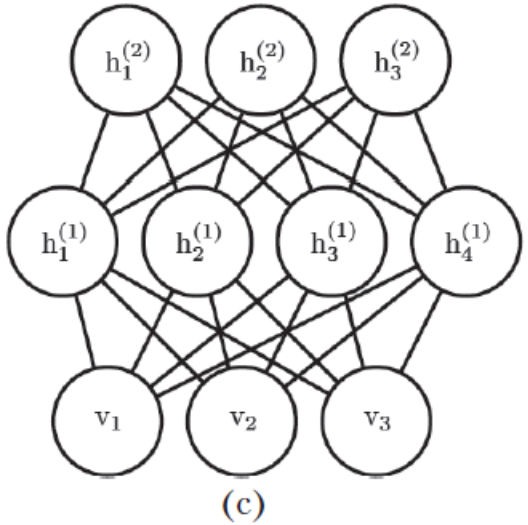
$$P(\mathbf{v} = \boldsymbol{v}, \mathbf{h} = \boldsymbol{h}) = (1/Z)\exp(-E(\boldsymbol{v}, \boldsymbol{h}))$$

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h}$$

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\}$$

УСЛОВНЫЕ РАСПРЕДЕЛЕНИЯ





УСЛОВНЫЕ РАСПРЕДЕЛЕНИЯ И ОБУЧЕНИЕ ОГРАНИЧЕННЫХ МАШИН БОЛЬЦМАНА

$$
\begin{aligned}
P(\boldsymbol{h} \mid \boldsymbol{v}) &= \frac{P(\boldsymbol{h}, \boldsymbol{v})}{P(\boldsymbol{v})} \\
&= \frac{1}{P(\boldsymbol{v})} \frac{1}{Z} \exp\{\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}\} \\
&= \frac{1}{Z'} \exp\{\boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}\} \\
&= \frac{1}{Z'} \exp\left\{ \sum_{j=1}^{n_h} c_j h_j + \sum_{j=1}^{n_h} \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j \right\} \\
&= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\{c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j\}.
\end{aligned}
$$

$$
\begin{aligned}
P(h_j = 1 \mid \boldsymbol{v}) &= \frac{\tilde{P}(h_j = 1 \mid \boldsymbol{v})}{\tilde{P}(h_j = 0 \mid \boldsymbol{v}) + \tilde{P}(h_j = 1 \mid \boldsymbol{v})} \\
&= \frac{\exp\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\}}{\exp\{0\} + \exp\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\}} \\
&= \sigma(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}).
\end{aligned}
$$

$$
P(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{n_h} \sigma((2\boldsymbol{h} - 1) \odot (\boldsymbol{c} + \boldsymbol{W}^\top \boldsymbol{v}))_j
$$

$$
P(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{n_v} \sigma((2\boldsymbol{v} - 1) \odot (\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}))_i
$$

Лекция 3.4 - Глубокие порождающие модели

ГЛУБОКИЕ СЕТИ ДОВЕРИЯ

$$P(\boldsymbol{h}^{(l)}, \boldsymbol{h}^{(l-1)}) \propto \exp(\boldsymbol{b}^{(l)\top}\boldsymbol{h}^{(l)} + \boldsymbol{b}^{(l-1)\top}\boldsymbol{h}^{(l-1)} + \boldsymbol{h}^{(l-1)\top}\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l)}),$$

$$P(h_i^{(k)} = 1 \mid \boldsymbol{h}^{(k+1)}) = \sigma(b_i^{(k)} + \boldsymbol{W}_{:,i}^{(k+1)\top}\boldsymbol{h}^{(k+1)})\forall i, \forall k \in 1, \ldots, l-2,$$

$$P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma(b_i^{(0)} + \boldsymbol{W}_{:,i}^{(1)\top}\boldsymbol{h}^{(1)})\forall i.$$

В случае вещественных видимых блоков подставляем

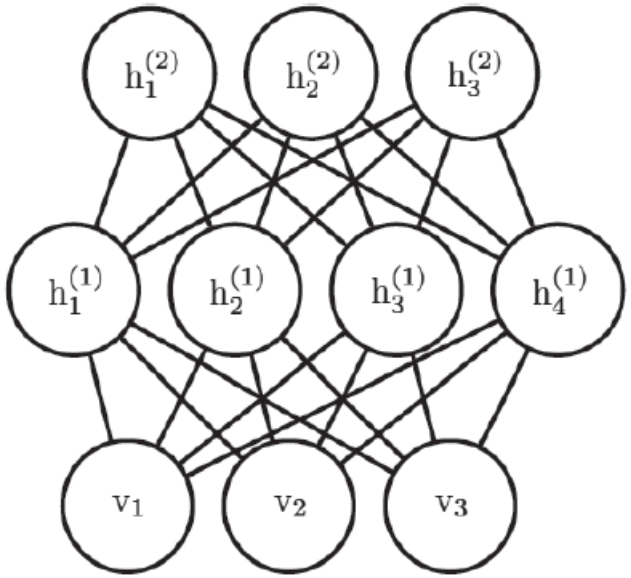
$$\mathbf{v} \sim \mathcal{N}(\boldsymbol{v}; \boldsymbol{b}^{(0)} + \boldsymbol{W}^{(1)\top}\boldsymbol{h}^{(1)}, \boldsymbol{\beta}^{-1}),$$

$$\mathbb{E}_{\mathbf{v}\sim p_{\text{data}}}\mathbb{E}_{\mathbf{h}^{(1)}\sim p^{(1)}(\boldsymbol{h}^{(1)}|\boldsymbol{v})} \log p^{(2)}(\boldsymbol{h}^{(1)})$$

$$\boldsymbol{h}^{(1)} = \sigma(b^{(1)} + \boldsymbol{v}^\top \boldsymbol{W}^{(1)}),$$

$$\boldsymbol{h}^{(l)} = \sigma(b_i^{(l)} + \boldsymbol{h}^{(l-1)\top}\boldsymbol{W}^{(l)})\forall l \in 2, \ldots, m.$$

ГЛУБОКИЕ МАШИНЫ БОЛЬЦМАНА



Графическая модель глубокой машины Больцмана с одним видимым слоем (внизу) и двумя скрытыми слоями. Связи существуют только между блоками из соседних слоев. Внутри слоев никаких связей нет.

$$P(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta})).$$

Лекция 3.4 - Глубокие порождающие модели

ПОСЛОЙНОЕ ПРЕДОБУЧЕНИЕ

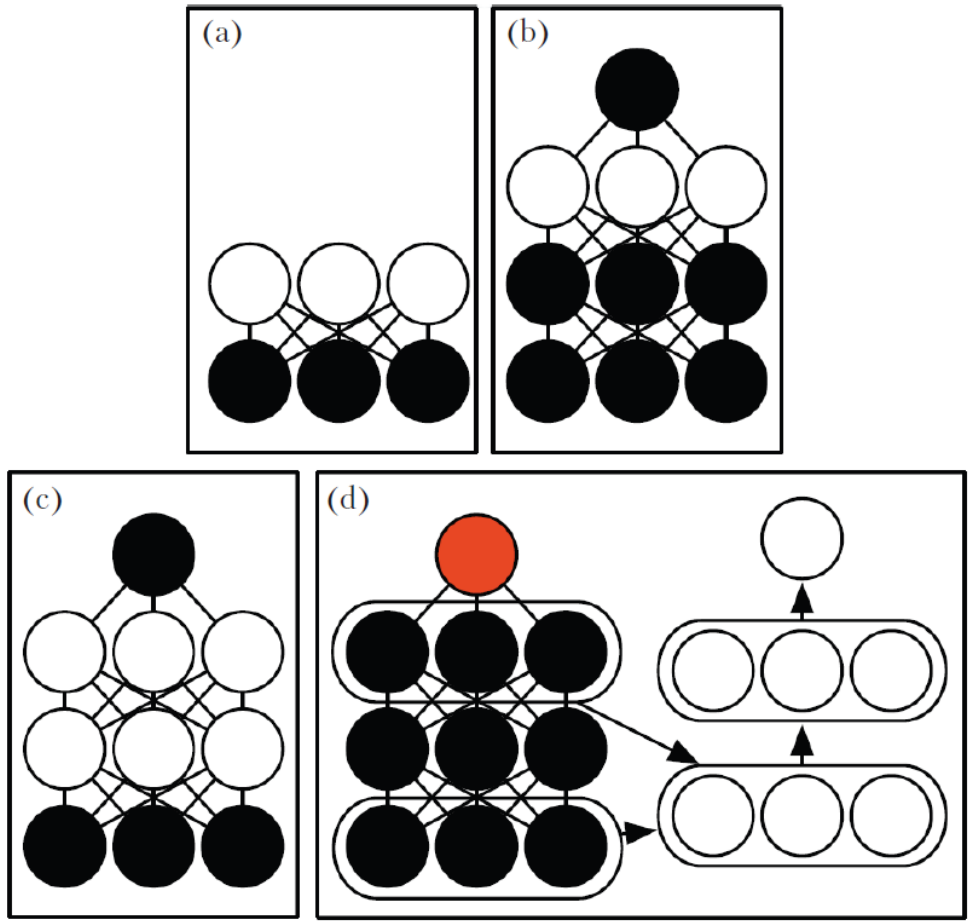
Алгоритм вариационной стохастической максимизации правдоподобия для обучения ГМБ с двумя скрытыми слоями

Установить размер шага ε равным малому положительному числу.
Установить число шагов выборки по Гиббсу k достаточно большим для приработки марковской цепи $p(v, h^{(1)}, h^{(2)}; \theta + \varepsilon\Delta_\theta)$.

Инициализировать три матрицы $\tilde{V}$, $\tilde{H}^{(1)}$ и $\tilde{H}^{(2)}$ с m строками каждая случайными значениями (например, выбранными из распределений Бернулли с такими же маргиналами, как у модели).
while не сошелся (цикл обучения) **do**
  Выбрать мини-пакет m примеров из обучающих данных и организовать его в виде строк матрицы плана V.
  Инициализировать матрицы $\hat{H}^{(1)}$ и $\hat{H}^{(2)}$, возможно, маргиналами модели.
  while не сошелся (цикл вывода среднего поля) **do**
    $\hat{H}^{(1)} \leftarrow \sigma(VW^{(1)} + \hat{H}^{(2)}W^{(2)\top})$
    $\hat{H}^{(2)} \leftarrow \sigma(\hat{H}^{(1)}W^{(2)})$
  end while
  $\Delta_{W^{(1)}} \leftarrow (1/m)V^\top\hat{H}^{(1)}$
  $\Delta_{W^{(2)}} \leftarrow (1/m)\hat{H}^{(1)\top}\hat{H}^{(2)}$
  for $l = 1$ to k (выборка по Гиббсу) **do**
    Блочная выборка по Гиббсу 1:
    $\forall i, j, \tilde{V}_{i,j}$ выбирается из $P(\tilde{V}_{i,j} = 1) = \sigma(W^{(1)}_{j,:}(\tilde{H}^{(1)}_{i,:})^\top)$.
    $\forall i, j, \tilde{H}^{(2)}_{i,j}$ выбирается из $P(\tilde{H}^{(2)}_{i,j} = 1) = \sigma(\tilde{H}^{(1)}_{i,:}W^{(2)}_{:,j})$.
    Блочная выборка по Гиббсу 2:
    $\forall i, j, \tilde{H}^{(1)}_{i,j}$ выбирается из $P(\tilde{H}^{(1)}_{i,j} = 1) = \sigma(\tilde{V}_{i,:}W^{(1)}_{:,j} + \tilde{H}^{(2)}_{i,:}W^{(2)\top}_{j,:})$.
  end for
  $\Delta_{W^{(1)}} \leftarrow \Delta_{W^{(1)}} - (1/m)V^\top\tilde{H}^{(1)}$
  $\Delta_{W^{(2)}} \leftarrow \Delta_{W^{(2)}} - (1/m)\tilde{H}^{(1)\top}\tilde{H}^{(2)}$
  $W^{(1)} \leftarrow W^{(1)} + \varepsilon\Delta_{W^{(1)}}$ (это упрощенная иллюстрация, на практике применяется более эффективный алгоритм, например импульсный с убывающей скоростью обучения)
  $W^{(2)} \leftarrow W^{(2)} + \varepsilon\Delta_{W^{(2)}}$
end while

ПОСЛОЙНОЕ ПРЕДОБУЧЕНИЕ

Процедура обучения глубокой машины Больцмана, использованной для классификации набора данных MNIST. (a) Обучить ОМБ, применив алгоритм CD для приближенной максимизации log P(v). (b) Обучить вторую ОМБ, которая моделирует $h^{(1)}$ и целевой класс y, применив алгоритм CD-k для приближенной максимизации log P(h(1), y), где $h^{(1)}$ – выборка из апостериорного распределения первой ОМБ при условии данных. Увеличивать k от 1 до 20 в процессе обучения. (c) Объединить обе ОМБ в ГМБ. Обучить ее приближенной максимизации log P(v, y), применив алгоритм стохастической максимизации правдоподобия с k = 5. (d) Удалить y из модели. Определить новый набор признаков $h^{(1)}$ и $h^{(2)}$, полученных путем выполнения вывода среднего поля в модели без y. Использовать эти признаки в качестве входа МСП, структура которого такая же, как структура дополнительного прохода среднего поля, с дополнительным выходным слоем для оценки y. Инициализировать веса МСП весами ГМБ. Обучить МСП приближенной максимизации log P(y | v), применив алгоритм стохастического градиентного спуска и прореживание.



САМОСТОЯТЕЛЬНАЯ РАБОТА

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
3	Математические прикладные аспекты глубокого обучения	Изучение учебной и научной литературы.	8 недель	50	Реферат	Рекомендованная учебная литература
3	Современные архитектуры глубоких нейронных сетей	Изучение учебной и научной литературы.	9 недель	62	Реферат	Рекомендован-ная учебная литература
Общая трудоемкость самостоятельной работы по дисциплине (час)				112		
Бюджет времени самостоятельной работы, предусмотренный учебным планом для данной дисциплины (час)				112		**133**