МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

БАЙЕСОВСКИЕ НЕЙРОННЫЕ СЕТИ

Курсовая работа

Афанасенко Григория Сергеевича
студента 2-го курса
специальности 1-31 03 09
«Компьютерная математика
и системный анализ»

Научный руководитель:
ст. преподаватель А. Э. Малевич

Минск, 2024

ОГЛАВЛЕНИЕ

ГЛАВА 1

Теоретические сведения.

1.1 Байесовский подход.

Перед тем, как приступить к сетям, вспомним основы байесовской статистики.

ГЛАВА 2

Виды нейронных сетей.

2.1 Детерминированные нейронные сети.

Сначала напомним, что такое обычные(детерминированные) нейронные сети и как они обучаются.

Основная задача обычных искусственных нейронных сетей(ANN) в том, чтобы аппроксимировать некоторую зависимость выхода y от входа x: $y = \Phi(x)$. Зависимость $\Phi(x)$ аппроксимируем через композицию последовательных преобразований.

Для простоты будем рассматривать обычные *полносвязные* сети со входом x, скрытыми(промежуточными) состояниями слоёв $\boldsymbol{h_i}$, функциями активации $a_i(\cdot)$ и выходом y:

$$\boldsymbol{h_0} = \boldsymbol{x}$$

$$\boldsymbol{h_i} = a_i(\boldsymbol{W_i} \cdot h_{i-1} + \boldsymbol{b_i}), i = \overline{1...n}$$

$$\boldsymbol{h_n} = \widehat{y}$$

$$L = \mathcal{L}(\widehat{y}, y),$$

где $\mathcal{L}(\cdot, \cdot)$ - функция ошибки.

Обозначим параметры модели на i-ом слое $\boldsymbol{\theta_i} = (\boldsymbol{W_i}, \boldsymbol{b_i})$, а параметры всей модели через $\boldsymbol{\Theta} = \{\boldsymbol{\theta_i} : \ i \ = \ \overline{1...n}\}$. Чаще всего нейронные сети принято рассматривать, как вычислительный граф/граф вычислений. Такой подход удобен с инженерной точки зрения, поскольку позволяет воспользоваться инструментом автоматического дифференцирования, и используется во всех современных фреймоворках: PyTorch, TensorFlow и прочие. Граф вычислений является ациклическим ориентированным графом, составленным из вершин-переменных и вершин-операций(Рисунок 2.1).

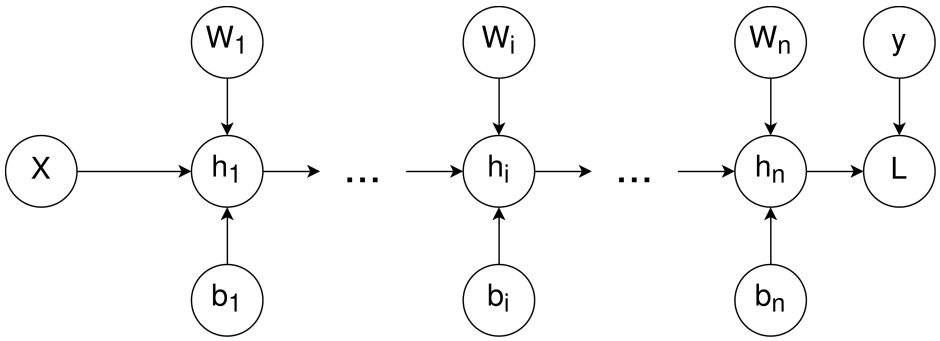

Рисунок 2.1 *Полносвязная* сеть в виде графа вычислений

Далее будем называть модели, основанные на графах вычислений, — графовыми моделями. Графы вычислений могут разных типов: статическими/динамическими, детерминированными/вероятностными и т.д. Для обучения/настройки параметров детерминированных графовых моделей используется метод *обратного распространения ошибки(back propagation)*, который широко используется в современном мире. Вкратце напомним алгоритм:

После прямого выполнения графа(*forward pass*), то есть в соответствии с направлениями рёбер на выходе мы получаем L -значение функции ошибки, которые в зависимости от задач мы хотим либо минимизировать, либо максимизировать. Для этого мы пользуемся градиентными методами оптимизации, что требует вычисление градиентов $\frac{dL}{dW_i}, \frac{dL}{db_i}$ по нашим параметрам модели, где $i = \overline{1,n}$. В общем случае это трудная задача, однако в случае детерминированных графовых моделей мы можем использовать цепное правило(*chain rule*) для того, чтобы последовательно проталкивать градиенты, начиная с концевой вершины, содержащей L.

Например, для подсчёта градиентов $\frac{dL}{dW_n}, \frac{dL}{db_n}$ мы представим его в виде

$$\frac{dL}{dW_n} = \frac{dL}{dh_n} \cdot \frac{dh_n}{dW_n}$$

$$\frac{dL}{db_n} = \frac{dL}{dh_n} \cdot \frac{dh_n}{db_n}$$

Аналогично для всех остальных параметров модели мы будем проталкивать накопленный с концевой вершины градиент до соответствующих вершин и с помощью этого градиента высчитывать градиент по параметрам модели. Схему работы алгоритма обратного распространения ошибки можно увидеть на Рису-

нок 2.2.

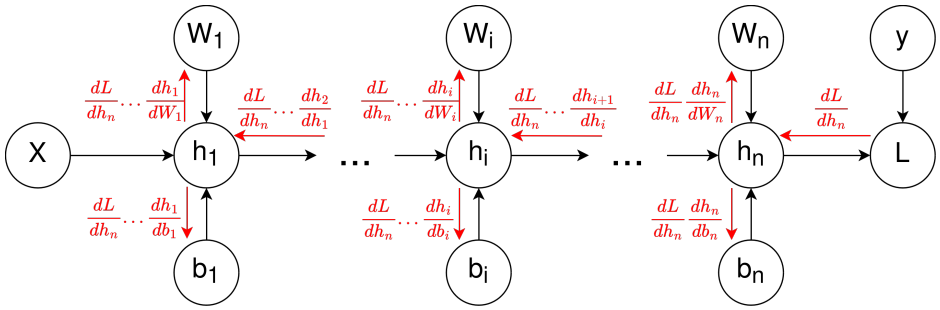

Рисунок 2.2 Обратное распространение ошибки по графу вычислений детерминированной полносвязной сети

Однако детерминированные нейронные сети обладают несколькими проблемами:

- Переобучение.
- Низкая интерпретируемость.
- Завышенная/заниженная уверенность модели в предсказаниях, даже если они неверные.
- Низкий уровень откалиброванности модели.

Указанные проблемы попытаемся решить с помощью байесовского подхода к нейронным сетям, который рассмотрим далее.

2.2 Байесовские нейронные сети.

2.2.1 Вероятностные графы вычислений.

Перед тем, как приступить к байесовским нейронным сетям, рассмотрим *вероятностные графы вычислений*, на которых основаны байесовские сети. В литературе также часто вместо названия *вероятностные графы вычислений* встречается *вероятностные графические модели*. Второе название является более общим, в то время как первое более специфично именно для байесовских нейронных сетей. Такие графы вычислений широко используются и известны достаточно давно. Они лежат в основе, например, Марковских цепей, которые

ранее активно использовались в различных задачах машинного предсказания, распознавания образов и т.п.

Основная мотивация в использовании вероятностного подхода состоит в том, что в реальном мире мы чаще имеем дело с неопределённостью в данных и знаниях и не можем детерминированно описать все приходящие переменные для решения задачи. Для решения проблем с неопределённостью можно попробовать собрать большие объёмы данных для того, чтобы попытаться "понять"эту неопределённость. С другой стороны мы можем использовать байесовский подход, который напрямую оперирует с неопределённостью.

Рассмотрим структуру *вероятностных графовых моделей*. В отличие от детерминированных моделей в граф добавляются вершины со случайными переменными. Таким образом в нашем совместно существуют детерминированные вершины и случайные (Рисунок 2.3). Стоит отметить, что после вступления в контакт детерминированных переменных и случайных, весь дальнейший результат будет случайным. При работе с такими моделями нужно различать *наблюдаемые* и *скрытые/латентные* переменные. Различия в этих двух понятиях естественны: в реальной жизни у нас есть некоторые известные данные и те, которые мы не может измерить явно, а лишь вычислить в результате работы модели.

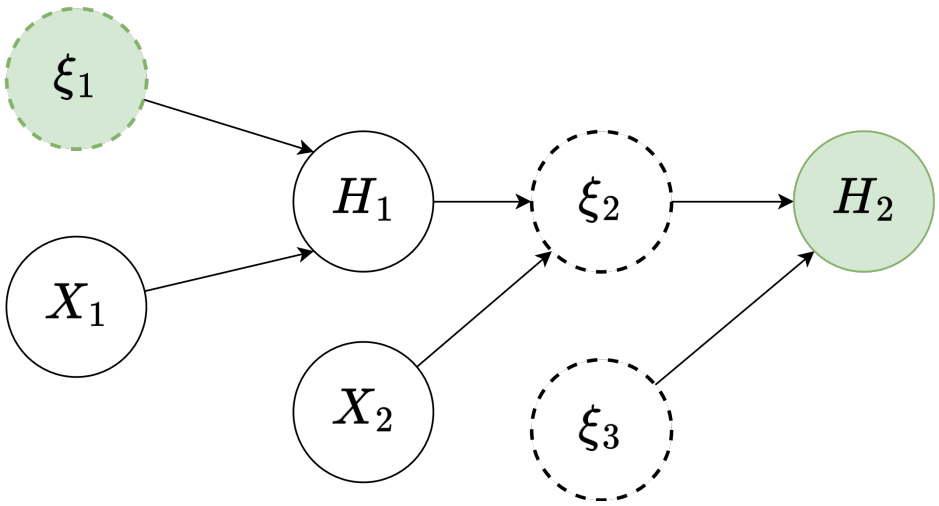

Рисунок 2.3 Вероятностная графическая модель. Здесь круги с пунктирной границей являются сэмплируемыми случайными величинами. Зелёным цветом обозначены наблюдаемые случайные переменные.

Стоит сделать замечание, что детерминированные переменные также можно

представить, как случайные величины с δ-функцией плотности распределения $\delta(\cdot)$, где $\delta(\cdot)$ — δ-функция Дирака. Данный факт позволяет рассматривать все вершины в вероятностной графовой модели, как случайные.

Введём более строгое определение. Пусть $(x_1, x_2, ..., x_n)$ - множество случайных величин, представляющих вершины ориентированного графа. Тогда *вероятностная графическая модель* — это семейство условных распределений $p(x_1|...)$, $p(x_2|...)$ и т.д. над данными случайными величинами $x_1, x_2, x_3, ..., x_n$.

В случае графовых моделей каждая случайная величина x_i зависит не от всех других случайных величин, а лишь от некоторого множество её предков $ancestors(x_i)$. Таким образом мы можем вычислить полную условную плотность величины x_i так:

$$p(x_i|x_n, x_{n-1}, ...x_1) = p(x_i|ancestors(x_i))$$

Используя *цепное правило* для совместного распределения $p(x_1, x_2, ..., x_n)$ мы можем расписать его через частные распределения и условные:

$$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)...p(x_n|x_{n-1}, ..., x_1)$$

Выбирая порядок множителей справа удобным образом мы можем вычислить совместное распределение.

Подобные *вероятностные графические модели* позволяют узнавать неочевидные взаимосвязи в данных, если в качестве вершин принять, например, признаки из какого-нибудь набора данных. При достаточном времени, потраченном на составлении связей в данном графе, аналитик данных способен в удобной форме отлавливать закономерности и проверять гипотезы о распределении данных. Также возможно их использование в системном или бизнес анализах, однако придётся потратить больше времени для дизайна нашего графа, поскольку мы можем столкнуться с не числовыми вершинами, а, например, событийными.

Другая полезная особенность таких моделей в том, что вместо какого-то конкретного значения интересующей нас величины мы получаем её распределение(Рисунок 2.4). Это даёт сильно больше информации, чем одно значение и позволяет оценивать *риски*, связанные с этой величиной. Существует много задач, где определение рисков важнее какого-то одного ответа. Примеры: задача

кредитного скоринга, большинство задач по работе с финансами(определение стоимости ценных бумаг, курса валют и т.д.), задачи в области медицины и здравоохранения, транспорт на автопилоте и т.п.

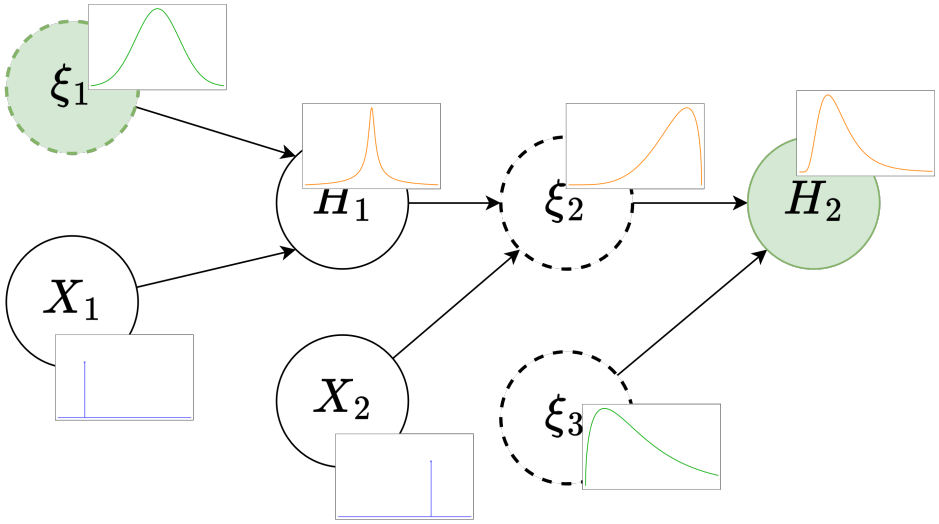

Рисунок 2.4 Та же графическая модель, но с видимыми распределениями значений в вершинах. Детерминированные вершины имеют δ-функцию распределения.

Существуют несколько инструментов для работы с такими моделями: Bayes Net Toolbox (`MATLAB`), pgmpy (`Python`)

2.2.2 Байесовские нейронные сети.