

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, 2025

# ОГЛАВЛЕНИЕ

<b>1</b>	<b>Теоретические сведения.</b>	<b>3</b>
1.1	Байесовский подход. . . . .	3
1.1.1	Основы байесовской статистики. . . . .	3
1.1.2	Байесовский подход к оценке параметров. . . . .	4
1.2	Вариационный вывод. . . . .	6
<b>2</b>	<b>Виды нейронных сетей.</b>	<b>7</b>
2.1	Детерминированные нейронные сети. . . . .	7
2.2	Байесовские нейронные сети. . . . .	9
2.2.1	Вероятностные графы вычислений. . . . .	9
2.2.2	Байесовские нейронные сети. . . . .	12

# ГЛАВА 1

## Теоретические сведения.

### 1.1 Байесовский подход.

#### 1.1.1 Основы байесовской статистики.

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

Байесовская статистика противопоставляется частотной статистике, как альтернатива. Основное отличие в двух подходах состоит в том, что в байесовской статистике вероятность интерпретируется, как степень уверенности в истинности суждения. Другими словами, как мера незнания или неопределённости. В частотной статистике вероятность определяется как частота события. Байесовскую вероятность ещё иногда называют «логической» вероятностью, поскольку её проще применять в реальных задачах.

Байесовский подход же к оценке параметров заключается в утверждении, что априорные знания влияют на апостериорные знания. Данное утверждение наиболее ярко видно в формуле Байеса:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} = \frac{P(D|H)P(H)}{\int_{\mathcal{H}} P(D|\tilde{H})P(\tilde{H})d\tilde{H}}$$
$$P(H|D) \propto P(D|H)P(H)$$

В данной формуле,

$H$  — некоторая гипотеза, вероятность которой мы хотим узнать, при помощи известных данных  $D$ .

$P(H|D)$  — апостериорная вероятность гипотезы после того, как мы пронаблюдали данные  $D$ .

$P(H)$  — это априорная вероятность или априорные знания о нашей гипотезе, которые мы знаем до того, как пронаблюдали данные  $D$ .

$P(D|H)$  — плотность распределения, которое называется правдоподобием наблюдаемых данных  $D$ , если гипотеза верна.

$P(D)$  — можно проинтерпретировать, как шум в данных. Особенно хорошо это заметно, когда мы записываем его в интегральной форме, где, как-бы, перебираем возможную гипотезу  $\tilde{H}$ , которая должна наилучшим образом объяснить наши данные.

Нижняя запись, опуская знаменатель, который не зависит от  $H$ , обозначает пропорциональную зависимость между апостериорной вероятностью и априорной вероятностью.

Байесовские статические методы использует Теорему Байеса для вычисления и обновления вероятности после получения новых данных.

### 1.1.2 Байесовский подход к оценке параметров.

Теперь перейдем к задаче оценке параметров статистических(вероятностных) моделей. К таким моделям можно отнести почти все модели машинного обучения, нейронные сети, марковские цепи и др.

В общем случае обозначим за  $a_\theta(x)$  — статистическую модель из параметризованного семейства  $\{a_\theta(x) : \theta \in \Theta\}$ , где  $\theta$  — параметры модели. В случае линейной регрессии  $y = w^T x + b$ ,  $\theta = \{w, b\}$ ; для нейронных сетей  $\theta$  — веса промежуточных слоёв; для марковских цепях  $\theta$  — вероятности на рёбрах и т.д.

Когда мы занимаемся выбором модели  $a_\theta(x)$ , которая наилучшим образом(в некотором смысле) описывают неизвестную закономерность в данных, то мы занимаемся подбором параметров  $\theta$ . В классическом подходе мы хотим найти *точечную оценку* на параметры  $\theta$ , то есть найти значения параметров  $\hat{\theta}$ , при котором качество нашей статистической модели будет наилучшим, т.е.  $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \operatorname{loss}_{D_y, D_x}(\theta)$ . Тут важно отметить, что нас интересует единственное такое оптимальное значение, даже если их может быть несколько. В этом и заключается точечный подход к оценке параметров.

Однако такой подход ничего не говорит про устойчивость найденного решения или, на языке байесовской статистики, уровня уверенности в том, что найденные  $\hat{\theta}$  является наилучшими. Чтобы лучше понять, рассмотрим пример с линейной регрессией  $y = w^T x + b$  на двух ситуациях(Рисунок 1.1, Рисунок 1.2).

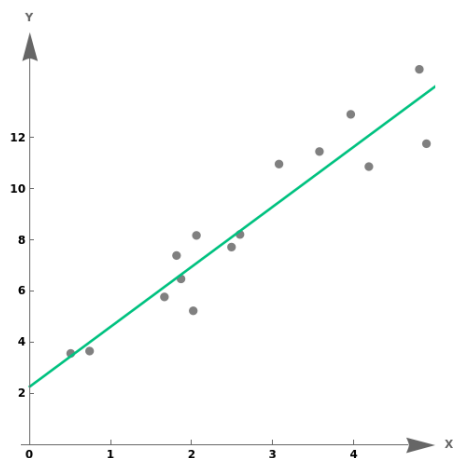


Рисунок 1.1 Уверенная модель

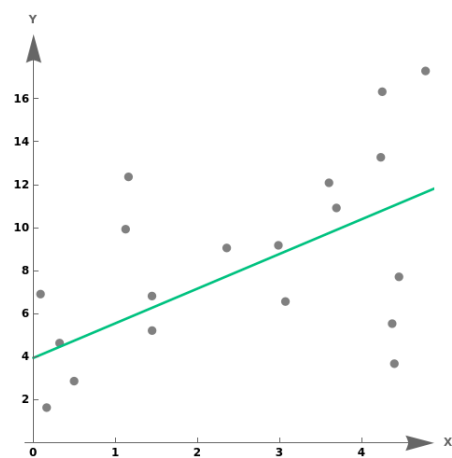


Рисунок 1.2 Неуверенная модель

И в первом, и во втором случае мы нашли оптимальную оценки  $\hat{w}, \hat{b}$ , однако во втором случае в данных сильно больше шума, что увеличивает неуверенность модели в том, что найденные оценки является наилучшими. Для того, чтобы лучше понимать ситуацию рассмотрим распределение параметров, а не их значение. Например, для примеров выше распределения параметров могли быть следующими:

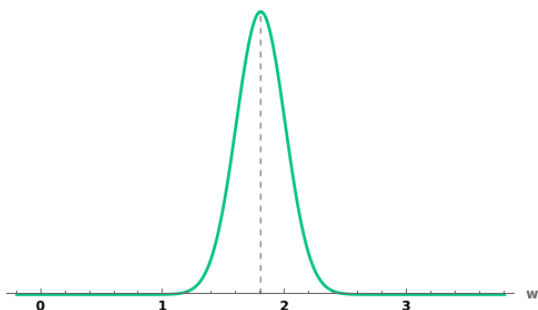


Рисунок 1.3 Уверенная модель



Рисунок 1.4 Неуверенная модель

Таким образом, одного лишь значения оценки параметров  $\hat{\theta}$  может быть недостаточно для того, чтобы правильно решить поставленную задачу. Гораздо больше информации даёт распределение параметров  $p(\theta)$ , поэтому в байесовском подходе **оценивается не сами параметры, а их распределение**.

Далее мы применим основную байесовскую парадигму и добавим априорные знания в нашу статическую модель. Вернёмся к точечной оценке параметров:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \operatorname{loss}_{D_y, D_x}(\theta)$$

Чаще всего в качестве  $loss_{D_y, D_x}(\cdot)$  предполагается брать  $-\log p(D_y|D_x, \theta)$  и получаем:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}(-\log p(D_y|D_x, \theta)) = \underset{\theta}{\operatorname{argmax}} \log p(D_y|D_x, \theta)$$

Получившаяся оценка  $\hat{\theta}$  называется *оценкой максимума правдоподобия (MLE)*. В данном случае  $\theta$  рассматривается, как неизвестный, но неслучайный параметр.

Теперь будем рассматривать  $\theta$ , как случайный параметр. Тогда будем искать значение  $\theta$ , которое максимизирует апостериорное распределение  $p(\theta|D_y, D_x)$  (или его логарифм, что тоже самое):

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}(\log p(\theta|D_x, D_y)) = \underset{\theta}{\operatorname{argmax}}(\log p(D_y|\theta, D_x)p(D_x|\theta)p(\theta) - \log p(D_y, D_x))$$

$$\hat{\theta} = \left[ p(D_x|\theta) = p(D_x), \text{ т.к. } D_x \perp \theta \right] = \underset{\theta}{\operatorname{argmax}}(\log p(D_y|\theta, D_x)p(D_x)p(\theta))$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(\log p(D_y|\theta, D_x))}_{-\mathcal{L}_{D_y, D_x}(\theta)} + \underbrace{\log p(\theta)}_{-\mathcal{R}(\theta)}$$

Таким образом мы получили *оценку апостериорного максимума (MAP)* на параметры  $\theta$ . Стоит заметить, что разница между итоговым оптимизируемым функционалом в случае *MLE* и случае *MAP* заключается в добавлении функционала  $\mathcal{R}(\theta)$ , который выступает в роли *регуляризатора* весов.

Таким образом добавление априорных знаний или априорного распределения параметров также добавляет естественный регуляризатор в функционал. Следовательно, просто меняя подход на байесовский наша модель становится более устойчива к переобучению.

**Замечание.** В будущем мы ещё вернёмся к нашему оптимизируемому функционалу, добавляя в него дополнительные слагаемые и тем самым получая новые свойства. Здесь также можно проследить интересный переход от вероятностной постановки задачи к оптимизационной, которую мы решаем с помощью градиентных методов.

## 1.2 Вариационный вывод.

dfsdfs

# 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  $\mathbf{h}_i$ , функциями активации  $a_i(\cdot)$  и выходом  $y$ :

$$\begin{aligned}\mathbf{h}_0 &= \mathbf{x} \\ \mathbf{h}_i &= a_i(\mathbf{W}_i \cdot \mathbf{h}_{i-1} + \mathbf{b}_i), i = \overline{1...n} \\ \mathbf{h}_n &= \hat{y} \\ L &= \mathcal{L}(\hat{y}, y),\end{aligned}$$

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

Обозначим параметры модели на  $i$ -ом слое  $\boldsymbol{\theta}_i = (\mathbf{W}_i, \mathbf{b}_i)$ , 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 фреймворках: PyTorch, TensorFlow и прочие. Граф вычислений является ациклическим ориентированным графом, составленным из вершин-переменных и вершин-операций(Рисунок 2.1).



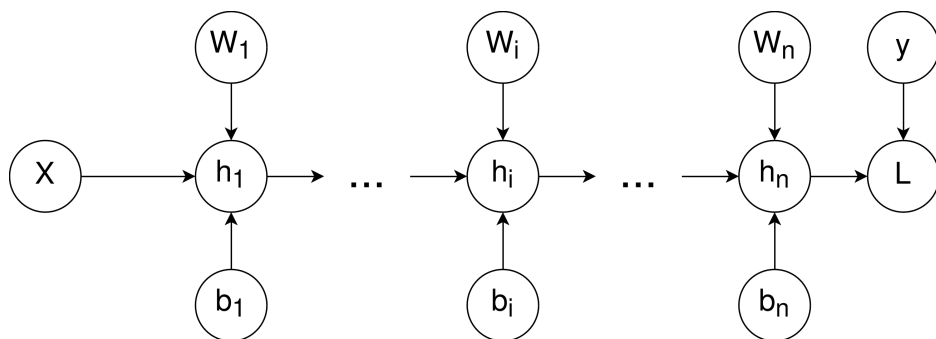


Рисунок 2.1 Полносвязная сеть в виде графа вычислений

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.

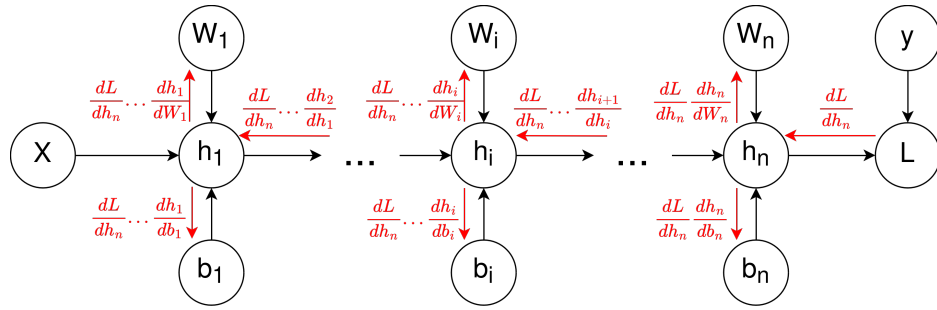


Рисунок 2.2 Обратное распространение ошибки по графу вычислений детерминированной полносвязной сети

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 можем измерить явно, а лишь вычислить в результате работы модели.

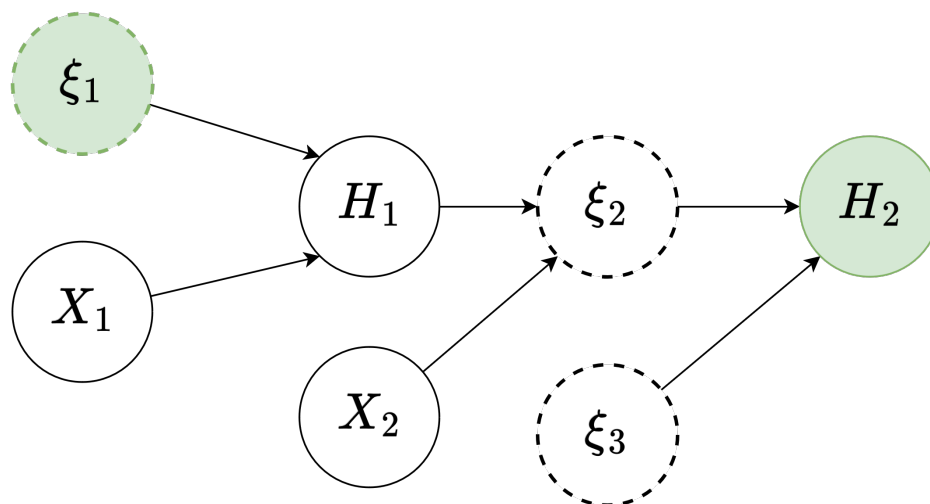


Рисунок 2.3 Вероятностная графическая модель. Здесь круги с пунктирной границей являются сэмплируемыми случайными величинами. Зелёным цветом обозначены наблюдаемые случайные переменные.

Стоит сделать замечание, что детерминированные переменные также можно

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

Введём более строгое определение. Пусть  $(x_1, x_2, \dots, x_n)$  - 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  $x_1, x_2, x_3, \dots, x_n$ .

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

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

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

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

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.

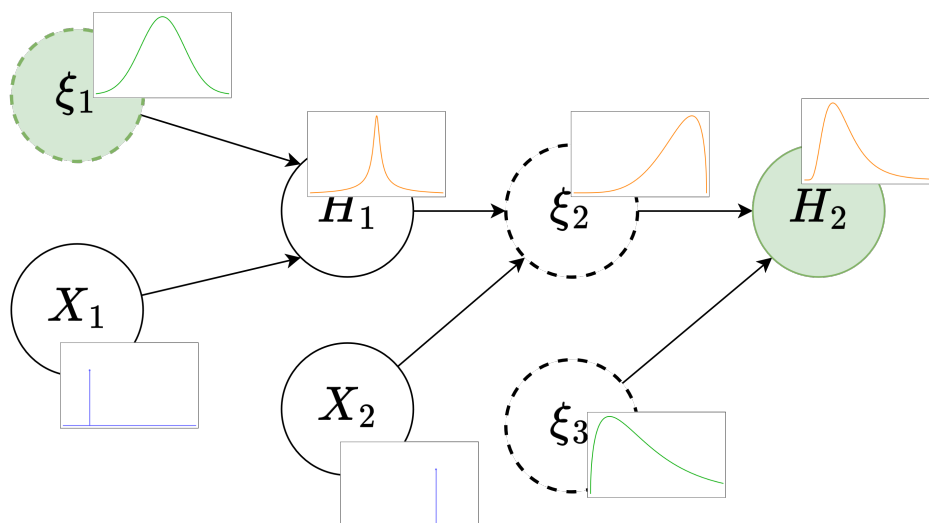


Рисунок 2.4 Та же графическая модель, но с видимыми распределениями значений в вершинах. Детерминированные вершины имеют  $\delta$ -функцию распределения.

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

## 2.2.2 Байесовские нейронные сети.