Машинное обучение (Machine Learning)

Mixture-of-Experts

Уткин Л.В.

ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

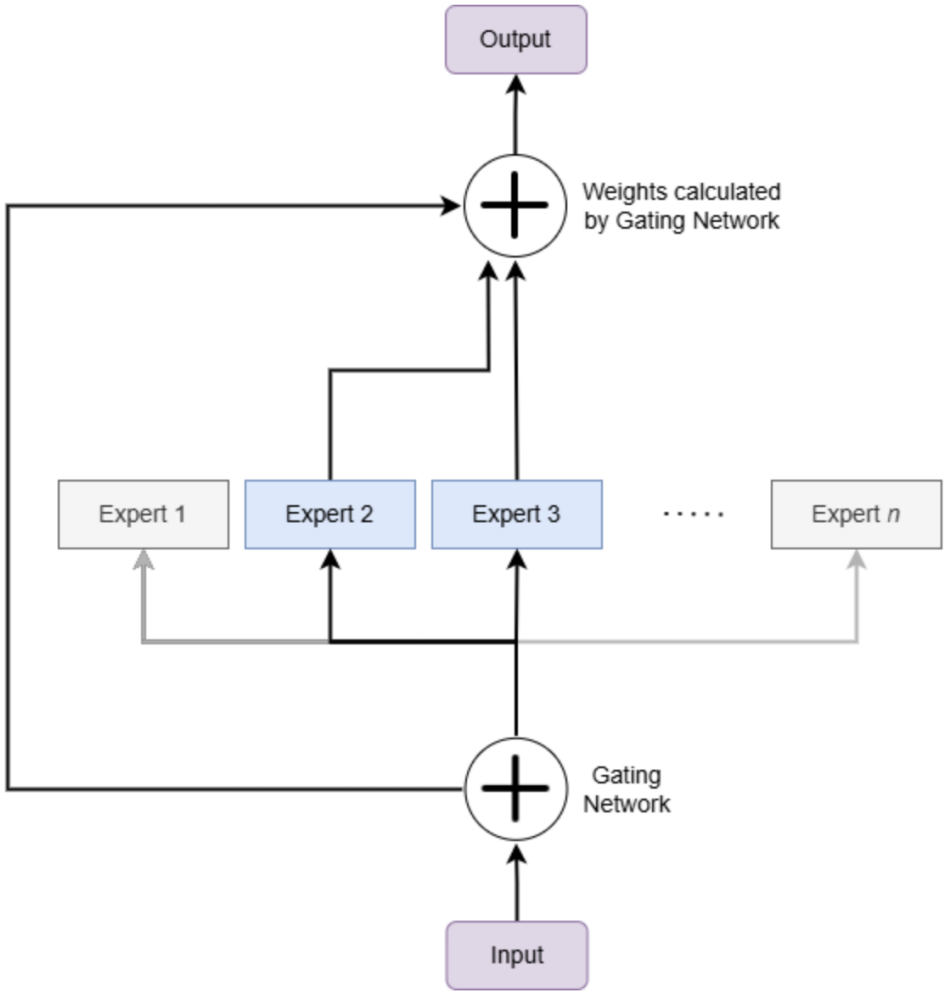
Mixture-of-Experts (MoE)

- **Идея**: Специализация моделей на разных частях входного пространства и наличие “менеджера”, который выбирает подходящего специалиста.
- **Цель**: Повышение эффективности модели. Вместо того чтобы заставлять одну гигантскую модель учить всю сложность данных, создается набор "экспертов каждый из которых становится специалистом в определенном типе данных.
- Модель вертикально масштабируется за счет увеличения числа экспертов, не увеличивая вычислительные затраты на один пример.

Принцип работы (1)

- *Два основных компонента*:
  - **Experts**: Отдельные модели (например, нейр. сети), которые специализируются на разных аспектах данных.
  - **Gating Network**: Одна модель (часто простая), которая обучается на лету определять, какому эксперту “доверить” текущий входной пример.
- Gating Network и Experts *обучаются* одновременно: Эксперт получает больше градиентов для обновления своих весов на тех примерах, где Gating Network его “выбрала”.
- *Ключевая особенность*: “Для каждой задачи — свой эксперт”.

Общая схема


Output
Weights calculated by Gating Network
Expert 1
Expert 2
Expert 3
.....
Expert n
Gating Network
Input

Принцип работы (2)

- Когда подается новое входное наблюдение на вход МоЕ, то:
- Gating Network анализирует входные данные и назначает веса (вероятности) каждому эксперту, например: [Эксперт 1: 0.8, Эксперт 2: 0.1, Эксперт 3: 0.1].
- Активируются (полностью или частично) только один или несколько экспертов с наибольшими весами. Остальные “молчат”.
- Итоговый прогноз — это взвешенная сумма прогнозов активированных экспертов.

Отличие от случайного леса (1)

- Mixture-of-Experts - это способ построить огромную и мощную модель, которая остается эффективной, потому что для каждой конкретной задачи задействует только свою небольшую часть. Он **вертикально масштабируется** за счет увеличения числа экспертов, не увеличивая вычислительные затраты на один пример.
- Случайный лес - это способ сделать стабильный и надежный прогноз, объединив множество простых и немного разных моделей. Он **горизонтально масштабируется** за счет количества моделей.

Аналогия для случайного леса

- Представьте, что задается сложный вопрос большой группе случайных прохожих (каждый из которых не эксперт).
- Мы усредняем их ответы, и за счет этого шум и ошибки отдельных людей компенсируются, давая в среднем более точный ответ, чем у одного случайного человека.

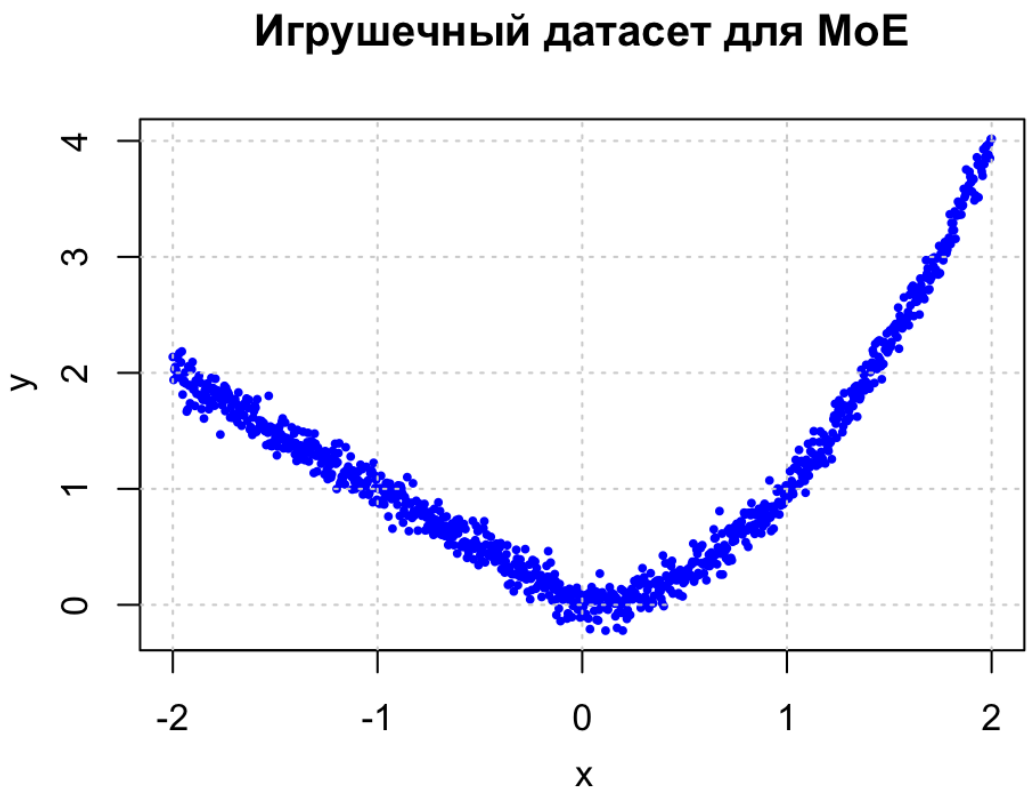
Аналогия для MoE

- Представим большую клинику.
- Вместо того чтобы каждый врач лечил все болезни (как в bagging), есть регистратор (Gating Network), который смотрит на ваши симптомы и направляет вас к конкретному специалисту (эксперту): кардиологу, неврологу или ортопеду.
- Получаете помощь от самого компетентного в вашей проблеме человека, а не от всех врачей сразу.

Пример (1)

- Пусть есть датасет, где целевая переменная y по-разному зависит от признака x на разных его участках:
  - при $x < 0$: $y = -x$
  - при $x \geq 0$: $y = x^2$
- Задача — обучить модель MoE, которая поймет это разделение и назначит правильных “экспертов” для каждого участка

Пример (2)



Архитектура MoE модели

- 2 эксперта (Expert Networks):
  - Expert 1: Обучаем линейную регрессию $y = w_1 \cdot x + b_1$
  - Expert 2: Обучаем полиномиальную регрессию $y = w_2 \cdot x^2 + b_2$
- Gating Network: используем простую логистическую регрессию, которая по x предскажет вероятности:
  - для Expert 1: $P(Expert = 1|x)$
  - для Expert 2: $1 - P(Expert = 1|x)$

Forward Pass

- Gating Network: вычисляет *веса* каждого эксперта на основе входного x:

$$g_1 = \sigma(w_g \cdot x + b_g), \quad g_2 = 1 - g_1$$

w_g, b_g - параметры Gating Network; g_1, g_2 - веса экспертов

- Expert Networks: каждый эксперт делает свое предсказание:

$$e_1 = w_1 \cdot x + b_1, \quad e_2 = w_2 \cdot x^2 + b_2$$

w_1, b_1 - параметры лин.эксперта; w_2, b_2 - квадр.эксперта

- Финальное предсказание - взвешенная сумма предсказаний экспертов:

$$y_{\text{pred}} = g_1 \cdot e_1 + g_2 \cdot e_2$$

Функция потерь

- Функция потерь - MSE:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(y_{\text{true}}^{(i)} - y_{\text{pred}}^{(i)} \right)^2$$

где N - количество примеров в обучающей выборке

- Градиент по итоговому предсказанию:

$$\frac{\partial L}{\partial y_{\text{pred}}} = \frac{2}{N} \sum_{i=1}^{N} \left(y_{\text{pred}}^{(i)} - y_{\text{true}}^{(i)} \right)$$

Еще градиенты

- Градиенты для экспертов:

$$\frac{\partial L}{\partial e_1} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot g_1, \quad \frac{\partial L}{\partial e_2} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot g_2$$

- Градиенты для Gating Network:

$$\frac{\partial L}{\partial g_1} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot (e_1 - e_2),$$

$$\frac{\partial g_1}{\partial z_g} = g_1 \cdot (1 - g_1), \quad z_g = w_g \cdot x + b_g$$

$$\frac{\partial L}{\partial w_g} = \frac{\partial L}{\partial g_1} \cdot \frac{\partial g_1}{\partial z_g} \cdot x, \quad \frac{\partial L}{\partial b_g} = \frac{\partial L}{\partial g_1} \cdot \frac{\partial g_1}{\partial z_g}$$

Обновление параметров

$$w_1 \leftarrow w_1 - \eta \cdot \frac{\partial L}{\partial w_1}, \quad \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial e_1} \cdot x$$

$$b_1 \leftarrow b_1 - \eta \cdot \frac{\partial L}{\partial b_1}, \quad \frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial e_1}$$

$$w_2 \leftarrow w_2 - \eta \cdot \frac{\partial L}{\partial w_2}, \quad \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial e_2} \cdot x^2$$

$$b_2 \leftarrow b_2 - \eta \cdot \frac{\partial L}{\partial b_2}, \quad \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial e_2}$$

$$w_g \leftarrow w_g - \eta \cdot \frac{\partial L}{\partial w_g}, \quad b_g \leftarrow b_g - \eta \cdot \frac{\partial L}{\partial b_g}$$

Результаты

- После обучения:
  - Веса экспертов:
    - Expert 1: (w_1, b_1) научится аппроксимировать прямую линию $y \approx -x$ для $x < 0$
    - Expert 2: (w_2, b_2) научится аппроксимировать параболу $y \approx x^2$ для $x \geq 0$
- Gating Network:
  - Научится выдавать $g_1 \approx 1, g_2 \approx 0$ (выбирать Эксперта 1) при $x < 0$
  - Научится выдавать $g_1 \approx 0, g_2 \approx 1$ (выбирая Эксперта 2) при $x \geq 0$

Общий случай MoE (Прямой проход)

- Имеется: K экспертов: $E_1, \ldots, E_K$; входной вектор: $\mathbf{x} \in \mathbb{R}^d$; Gating Network: $G(\mathbf{x})$
- **Gating Network**: вычисляет **веса** для каждого эксперта:

$$\mathbf{g} = \text{Softmax}(\mathbf{W}_g \mathbf{x} + \mathbf{b}_g)$$

  - $\mathbf{W}_g \in \mathbb{R}^{K \times d}$ - матрица весов Gating Network
  - $\mathbf{b}_g \in \mathbb{R}^K$ - вектор смещений
  - $\mathbf{g} = [g_1, \ldots, g_K]^T$ - вектор весов экспертов $\sum_{k=1}^{K} g_k = 1,\ g_k \geq 0$

Общий случай MoE (Прямой проход)

- Каждый **эксперт** делает свое предсказание:

$$e_k = E_k(\mathbf{x}) \text{ для } k = 1, \ldots, K$$

где E_k может быть нейронной сетью, линейной моделью, деревом решений

- Итоговое предсказание

$$y_{\text{pred}} = \sum_{k=1}^{K} g_k \cdot e_k$$

Механизм работы Gating Network

- **Softmax** обеспечивает **конкуренцию** между экспертами:

$$g_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}, \quad z_k = \mathbf{w}_g^{(k)} \cdot \mathbf{x} + b_g^{(k)}$$

  где $\mathbf{w}_g^{(k)}$ – k-я строка матрицы $\mathbf{W}_g$
- **Интерпретация весов** g_k: **вероятность** того, что эксперт k является подходящим для входа $\mathbf{x}$
- **Разделение пространства признаков:** Gating Network учится разделять входное пространство на регионы, где разные эксперты являются “специалистами”:

$$\text{Регион}_k = \{\mathbf{x} \in \mathbb{R}^d : g_k(\mathbf{x}) > g_j(\mathbf{x}),\ \forall j \neq k\}$$

Механизм обучения

- Gating Network учится через **совместную оптимизацию** с экспертами. Рассмотрим градиенты:

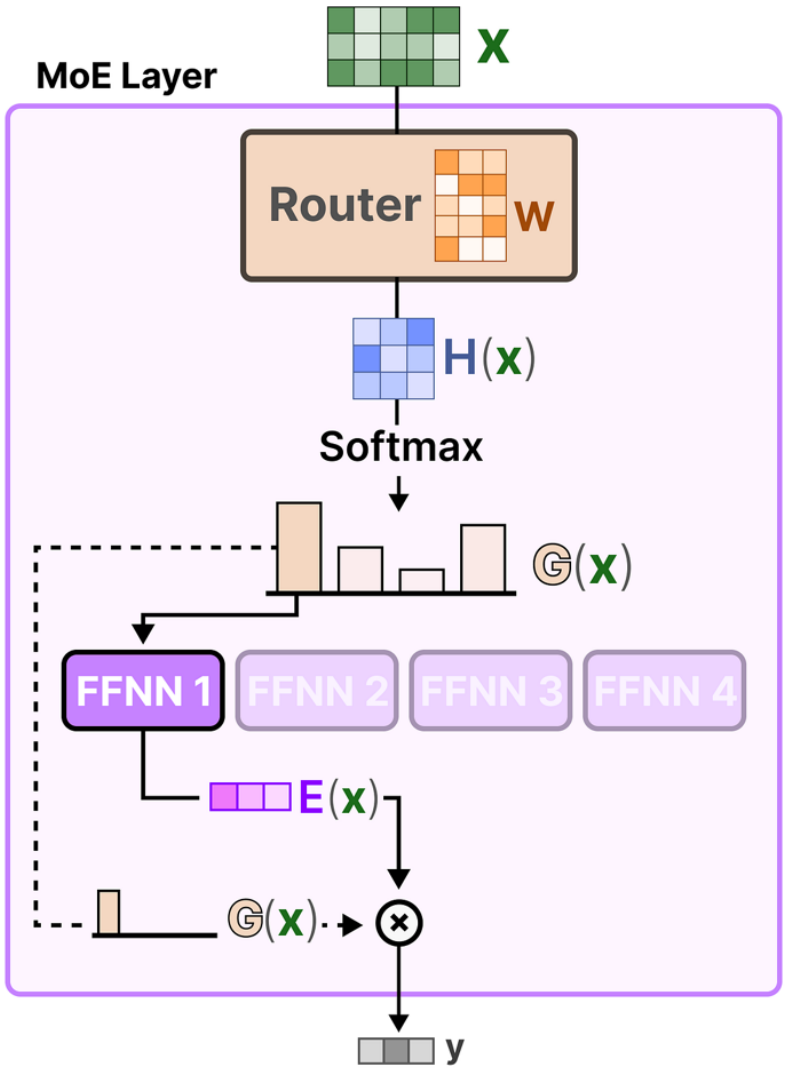
$$\frac{\partial L}{\partial \mathbf{w}_g^{(k)}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial g_k} \cdot \frac{\partial g_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_g^{(k)}}$$
$$= (y_{\text{true}} - y_{\text{pred}}) \cdot (e_k - y_{\text{pred}}) \cdot g_k(1 - g_k) \cdot \mathbf{x}$$

- Ключевой член: $(e_k - y_{\text{pred}})$ - **разница в качестве** предсказаний.

Принцип "проб и ошибок" через градиенты

- Пусть есть 3 эксперта и Gating Network их распределяет
- В начале обучения:
  - Все эксперты одинаково плохи во всех задачах и Gating Network случайно распределяет задачи
  - Из-за случайных нач. значений один эксперт может случайно оказаться чуть лучше в определенных задачах
  - Когда менеджер видит, что эксперт A хорошо справился с задачей типа X, эксперты B и C справились плохо, он через градиенты увеличивает вес эксперта A для задач типа X.

Механизм работы Gating Network (Router)



Gating Network и Router

- **Gating Network** - классический подход, где все эксперты участвуют с ненулевыми весами
- **Router** - современная оптимизация, где активируется только подмножество экспертов. Обеспечивает:
  - Вычислительную эффективность
  - Масштабируемость до тысяч экспертов
  - Практическую применимость в больших моделях
  - В современной литературе термины часто используются как синонимы, но технически Router - это спарсная версия Gating Network
  - В контексте современных LLM типа Mixtral, Switch Transformer и других, почти всегда используется Router с топ-k выбором

Gating Network и Router

- **Gating Network** (классический)

$$\mathbf{g} = \text{Softmax}(\mathbf{W}_g \mathbf{x}), \; y = \sum_{k=1}^{K} g_k \cdot E_k(\mathbf{x})$$

где все $g_k > 0$ и $\sum g_k = 1$.

- **Router** (современный)

$$\mathbf{s} = \mathbf{W}_r \mathbf{x} \text{ (логиты)}, \; \text{top-k} = \text{TopK}(\mathbf{s}, k)$$

$$g_i = \begin{cases} \frac{\exp(s_i)}{\sum_{j \in \text{top-k}} \exp(s_j)} & \text{if } i \in \text{top-k} \\ 0 & \text{otherwise} \end{cases}, \; y = \sum_{i \in \text{top-k}} g_i \cdot E_i(\mathbf{x})$$

Ключевые принципы

- **Автоматическое разделение**: Gating Network автоматически находит оптимальное разделение входного пространства
- **Специализация**: Каждый эксперт становится специалистом в своем регионе
- **Конкуренция**: Softmax заставляет экспертов конкурировать за данные
- **Непрерывность**: Веса меняются плавно
- **Масштабируемость**: Можно добавлять новых экспертов для более сложных задач

Регрессия Надарая-Уотсона

- Обучающая выборка $\{\mathbf{x}_i, y_i\}$, $i = 1, ..., N$:

$$f_{NW}(\mathbf{x}) = \sum_{i=1}^{N} \alpha(\mathbf{x}, \mathbf{x}_i) \cdot y_i$$

- Вес внимания

$$\alpha(\mathbf{x}, \mathbf{x}_i) = \frac{K(\mathbf{x}, \mathbf{x}_i, w)}{\sum_{j=1}^{N} K(\mathbf{x}, \mathbf{x}_j, w)} = \mathrm{softmax}\left(-w \left\|\mathbf{x} - \mathbf{x}_i\right\|^2\right)$$

Нейрон. сеть как параметрическая регр. Н-У

- Выходной слой НС:

$$f_{NN}(\mathbf{x}) = \sum_{i=1}^{h} \phi\left(\mathrm{LN}\left(\langle \mathbf{w}_i, \Phi(\mathbf{x}) \rangle\right)\right) \cdot v_i$$

- ϕ - функция активации; $\Phi(\mathbf{x})$ - преобразование входа в выход; $\mathbf{V} = [v_1, ..., v_h]$ - веса выходного слоя; $\mathrm{LN}(\cdot)$ - слой нормализации
- Сравнивая $f_{NN}(\mathbf{x})$ с $f_{NW}(\mathbf{x})$, видим, что НС неявно определяет параметризованную функцию ядра через НС:

$$K(\mathbf{x}, \{\mathbf{w}_i, b_i\}) = \phi\left(\langle \mathbf{w}_i, \Phi(\mathbf{x}) \rangle\right)$$

где нормализация применяется после ядра, роль меток y_i играют векторы значений v_i, а Φ - функция преобразования

Нейрон. сеть как параметрическая регр. Н-У

- В этой аналогии нормализация в регрессии Н-У соответствует L_1-нормализации $\mathrm{LN}(\mathbf{x}) = \mathbf{x}/\left\|\mathbf{x}\right\|_1$.
- Естественное обобщение - замена нормализации $\mathrm{LN}(\mathbf{x})$ в регрессии Н-У на более распространенную L_2-нормализацию в НС.
- Это дает математическую интерпретацию НС как частного случая параметрической регрессии Н-У.

MoE как параметрическая регр. Н-У

- MoE комбинирует Expert Networks $\{E_m(\mathbf{x})\}_{m=1}^{M}$ через Router $\mathbf{g}(\mathbf{x})$:

$$\mathrm{MoE}(\mathbf{x}) = \sum_{m=1}^{M} g_m(\mathbf{x}) \cdot E_m(\mathbf{x})$$

где Router $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), ..., g_M(\mathbf{x})]$ использует выходы softmax:

$$g_m(\mathbf{x}) = \frac{\exp\left(\langle \mathbf{w}_m, \mathbf{x} \rangle\right)}{\sum_{j=1}^{M} \exp\left(\langle \mathbf{w}_j, \mathbf{x} \rangle\right)}$$

- Уравнение для $\mathrm{MoE}(\mathbf{x})$ похоже на регр. Н-У, где вес $g_m(\mathbf{x})$ рассматривается как ядро $K(\mathbf{x}.\mathbf{w}_m)$, а каждый экспертный выход $E_m(\mathbf{x})$ соот-вует y_m.

MoE как параметрическая регр. Н-У

- Если вернуться к НС, реализующей ядро, то MoE может быть реализована как большая НС, которая агрегирует Expert Networks $\{E_m(\mathbf{x})\}_{m=1}^{M}$ при помощи таких НС, реализующих ядро:

$$g_m(\mathbf{x}) = \phi\left(\mathrm{LN}\left(\langle \mathbf{w}_m, \Phi(\mathbf{x})\rangle\right)\right)$$

Функция Router (маршрутизатора)

- Пусть $\Phi(\mathbf{x}) \in \mathbb{R}^d$ - представление, которое поступает на Router.
- Вводим новую функцию маршрутизатора $g_m(\mathbf{x}) = \phi(\cdot)$ с нормализацией на основе ядра (KERN), который реализует Router с помощью НС с линейной проекцией, за которой следуют (i) L_2-нормализация, (ii) активация ReLU и (iii) оптимальный обучаемый оператор масшабирования:

$$s(\mathbf{x}) = \mathbf{W}_s\Phi(\mathbf{x}) + \mathbf{b}_s; \bar{s}(\mathbf{x}) = \frac{s(\mathbf{x})}{\|s(\mathbf{x})\|_2 + \varepsilon};$$
$$\mathbf{r}(\mathbf{x}) = \mathrm{ReLU}(\bar{s}(\mathbf{x})); \widehat{\mathbf{g}}(\mathbf{x}) = \gamma \cdot \mathbf{r}(\mathbf{x})$$

где $\mathbf{W}_s \in \mathbb{R}^{M\times d}$; $\mathbf{b}_s \in \mathbb{R}^M$ - параметры Router

Функция Router (маршрутизатора)

- Нормализация сохраняет масштаб логит-анализа инвариантным к числу экспертов M, в то время как
- Активация ReLU сохраняет разреженность без использования экспоненциальных функций.
- В процессе вывода и обучения сохраняем только k экспертов, прошедших Router.

Вопросы

?