КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ АЛЬ-ФАРАБИ · КАФЕДРА КОМПЬЮТЕРНЫХ НАУК

Научное машинное обучение

Лекция 3. Нейронные сети как аппроксиматоры функций; автоматическое дифференцирование

Многослойный перцептрон, универсальная аппроксимация и вычисление производных через autograd

Кенжебек Ержан Ғалымжанұлы

PhD, старший преподаватель кафедры компьютерных наук

Цель и план лекции

Цель

Изучить нейронные сети как аппроксиматоры функций и освоить автоматическое дифференцирование — основу вычисления невязок в физически-информированных моделях.

План

1. Многослойный перцептрон и функции активации
2. Теорема универсальной аппроксимации
3. Обучение: потери и оптимизаторы
4. Способы вычисления производных
5. Автоматическое дифференцирование в PyTorch
6. Производные высокого порядка и их ограничения

Многослойный перцептрон

Нейросеть — параметрическая функция $u_\theta(x)$, составленная из чередующихся линейных преобразований и нелинейностей.

$$u_\theta(x) = W_L \cdot \sigma(\dots \sigma(W_2 \cdot \sigma(W_1 x + b_1) + b_2) \dots) + b_L$$

Параметры $\theta = \{W_k, b_k\}$ подбираются по данным. Ширина и глубина определяют выразительность модели.

Функции активации

Функция	Формула	Для ДУЧП
tanh	(e^x − e^(−x)) / (e^x + e^(−x))	Гладкая, стандартный выбор для PINN
sin (SIREN)	sin(ωx)	Гладкая, хороша для высоких частот
Sigmoid	1 / (1 + e^(−x))	Гладкая, затухающие градиенты
ReLU	max(0, x)	Вторая производная равна нулю — не подходит

Для уравнений второго порядка активация должна быть дважды дифференцируемой.

Теорема универсальной аппроксимации

Сеть с одним скрытым слоем и неполиномиальной непрерывной активацией может приблизить любую непрерывную функцию на компактном множестве с заданной точностью (Цыбенко, 1989; Хорник, 1991).

Что гарантирует

Существование аппроксимирующей сети.

Чего не гарантирует

Число нейронов, скорость обучения и качество экстраполяции.

Поэтому на практике важны архитектура, оптимизация и физические ограничения.

Обучение нейросети

Подбор θ — минимизация функции потерь на обучающих данных (x_i, y_i).

$$L(\theta) = (1/N)\cdot\Sigma_i \left(u_\theta(x_i) - y_i \right)^2$$

- Градиентный спуск и стохастические варианты
- Adam — адаптивный шаг; стандарт для первой стадии
- L-BFGS — квазиньютоновский метод; часто для доводки
- Градиенты по параметрам вычисляются обратным распространением

Четыре способа вычислить производную

Способ	Точность	Стоимость и проблемы
Ручной вывод	Точно	Трудоёмко, ошибки
Символьное	Точно	Разбухание выражений
Конечные разности	Ошибка округления и усечения	Выбор шага h, дорого по параметрам
Автоматическое	Точно до машинной точности	Умеренная память и время

Вычислительный граф и правило цепочки

Программа раскладывается на элементарные операции; производная собирается по правилу цепочки вдоль графа.

Прямой режим

Производные передаются вместе со значениями. Эффективен при малом числе входов.

Обратный режим

Градиент выхода по всем входам за один обратный проход. Основа обучения: параметров много, выход один.

Обратный режим требует хранить промежуточные значения прямого прохода.

Производные сети по входу в PyTorch

```
x = torch.linspace(0, 1, 100).reshape(-1, 1)
x.requires_grad_(True)
u = net(x)
du = torch.autograd.grad(u, x, torch.ones_like(u),
create_graph=True)[0]
d2u = torch.autograd.grad(du, x, torch.ones_like(du),
create_graph=True)[0]
```

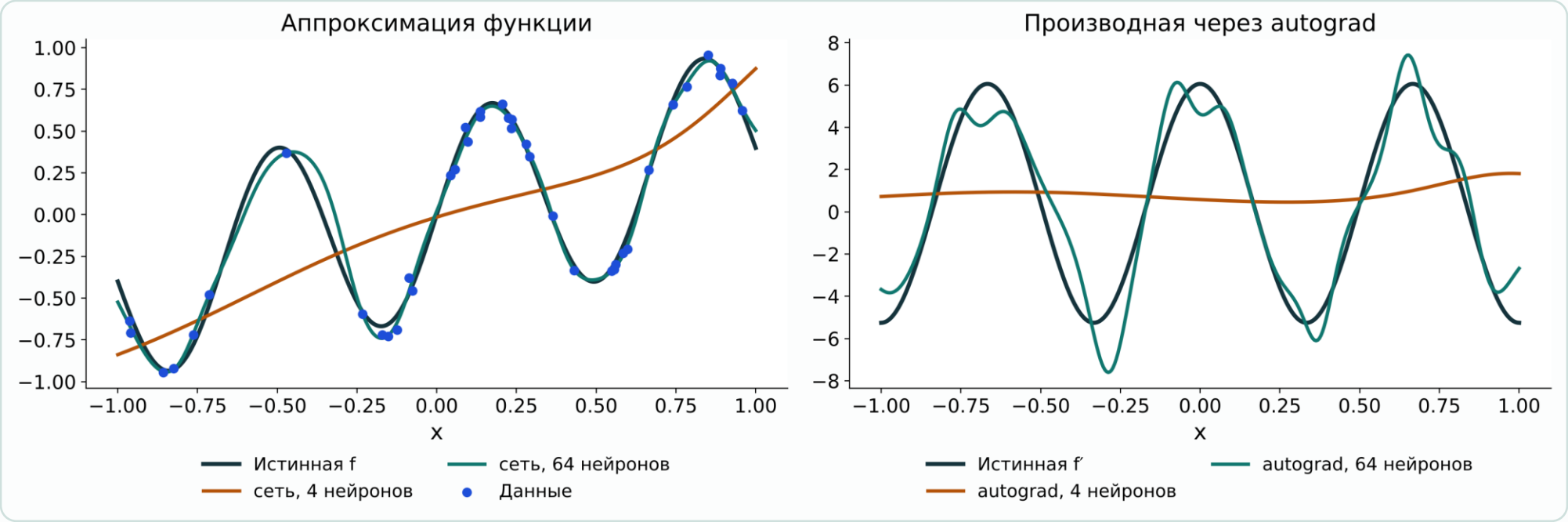
Параметр create_graph=True сохраняет граф, поэтому производные можно дифференцировать повторно и включать в функцию потерь.

Производные высокого порядка

- Для u_xx, u_t, u_xxx строится цепочка вызовов autograd.grad.
- Стоимость и память растут с порядком производной.
- Для векторных и многомерных задач используют якобианы и гессианы.
- Производная сети — гладкая функция только при гладкой активации.

Именно такие производные формируют невязку ДУЧП в PINN.

Сеть приближает функцию, но не производную



Зашумлённые данные f(x) = 0,6·sin(3πx) + 0,4x. Сеть из 64 нейронов хорошо приближает f, однако её производная через autograd заметно отклоняется от f′; сеть из 4 нейронов недообучена.

Мотивация физических ограничений

Данные не задают производные

Малое значение потерь по f не гарантирует точность f' и f''.

Нет законов

Сеть может нарушать уравнение между точками измерений.

Решение

Добавить в потери невязку уравнения — основа PINN.

Практические рекомендации

- Нормировать входы на отрезок $[-1; 1]$ или $[0; 1]$.
- Использовать tanh и инициализацию Ксавье.
- Для небольших задач применять float64.
- Сначала Adam, затем L-BFGS для уточнения.
- Проверять производные на функциях с известным ответом.

Выводы

- Нейросеть — гибкий параметрический аппроксиматор; универсальная аппроксимация гарантирует существование, но не обучаемость.
- Для ДУЧП нужны гладкие активации, например tanh.
- Автоматическое дифференцирование вычисляет производные точно, без разностных шагов.
- Производные сети по входу задаются через autograd.grad с create_graph=True.
- Совпадение по данным не гарантирует правильные производные — это мотивирует PINN.

Контрольные вопросы

1. Что утверждает теорема универсальной аппроксимации и каковы её ограничения?
2. Почему ReLU неудобна для PINN с уравнениями второго порядка?
3. Чем автоматическое дифференцирование отличается от численного и символьного?
4. В чём различие прямого и обратного режимов?
5. Что делает параметр create_graph=True?
6. Почему хорошая аппроксимация функции не гарантирует хорошую производную?

Литература

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.
2. Cybenko G. Approximation by superpositions of a sigmoidal function // Math. Control Signals Systems. 1989. Vol. 2. P. 303–314.
3. Hornik K. Approximation capabilities of multilayer feedforward networks // Neural Networks. 1991. Vol. 4. P. 251–257.
4. Baydin A.G. et al. Automatic differentiation in machine learning: a survey // JMLR. 2018. Vol. 18. P. 1–43.
5. Paszke A. et al. PyTorch: an imperative style, high-performance deep learning library // NeurIPS. 2019.
6. Kingma D.P., Ba J. Adam: a method for stochastic optimization // ICLR. 2015.