Практическая реализация ядерных моделей глубокого обучения

Максим Юрьевич Винниченко

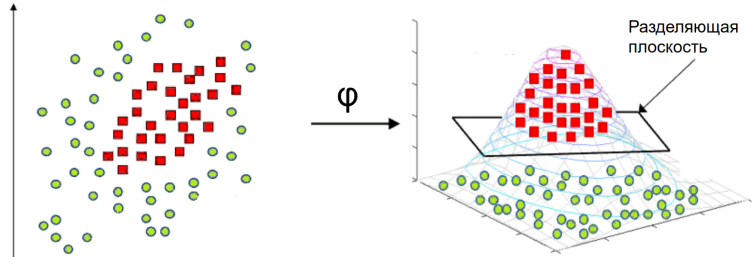
научный руководитель: И.Е. Куралёнок

НИУ ВШЭ

6 июня 2021 г.

Ядерный трюк

- Есть объекты X, которые плохо разделимы линейной моделью.
- Переведём объекты X в пространство $\varphi(\mathcal{X})$, где они хорошо разделяются линейной моделью.
- Для обучения линейной модели нужно уметь считать только функцию k "ядро"[1]: $k(z, x) = \langle \varphi(z), \varphi(x) \rangle$



[1] Craig Saunders, Alexander Gammerman и Volodya Vovk. "Ridge regression learning algorithm in dual variables". В: (1998).

Kernel Regression

Обучаем модель на объектах X и соответствующих ответах Y.
Обозначения: $K(Z,X) = \varphi(Z)\varphi(X)^T, \quad K = K(X,X)$
Матрицу K назовём матрицей ядра

- Обучение линейной модели:

$$\alpha = K^{-1}Y$$

- Предсказание линейной модели на объектах Z:

$$f(Z) = K(Z,X)\alpha$$

Связь с нейронными сетями

- Есть связь между широкими нейронными сетями и ядерными методами[2].
- Обучение бесконечно широкой сети – это обучение ядерной модели с ядром NTK[3]: $k(z, x) = \mathbb{E}_\theta \langle \nabla_\theta f(z, \theta), \nabla_\theta f(x, \theta) \rangle$.
- Обучение нейросетевого гауссовского процесса – это обучение ядерной модели с ядром NNGP: $k(z, x) = \mathbb{E}_\theta \, f(z, \theta) \cdot f(x, \theta)$

[2] Jaehoon Lee и др. “Wide neural networks of any depth evolve as linear models under gradient descent”. В: *Advances in neural information processing systems*. 2019, с. 8570—8581.

[3] Arthur Jacot, Franck Gabriel и Clément Hongler. “Neural tangent kernel: Convergence and generalization in neural networks”. В: *Advances in neural information processing systems*. 2018, с. 8571—8580.

Актуальность

- Аналитические ядерные методы приблизились к нейросетевым методам в задаче классификации изображений[4].
- Однако, они плохо масштабируются на большие обучающие выборки.
- Пусть N – размер обучающей выборки, P – число пикселей в изображении, тогда:
  - сложность вычисления ядра аналитически $\mathcal{O}(N^2P^2)$
  - сложность обращения ядра $\mathcal{O}(N^3)$

[4] Vaishaal Shankar и др. “Neural kernels without tangents”. В: *International Conference on Machine Learning*. PMLR. 2020, с. 8614—8623.

Цели и задачи

Цель: разработать масштабируемый нейросетевой ядерный метод для классификации изображений
Задачи:

- Обобщить метод Kernel Regression на случай необратимой матрицы
- Снизить сложность оптимизации двойственных переменных
- Построить приближение аналитического ядра случайными признаками
- Сравнить метод с нейронными сетями на наборах данных Tiny-Imagenet и Imagenet-R

Задача 1: случай необратимой матрицы

Существуют 2 аналитических подхода к вычислению двойственных переменных α в случае необратимой матрицы ядра K:

1. Kernel Ridge Regression[5]:

$$\alpha = (K + 1/\eta \cdot E)^{-1} Y$$

2. Обучение градиентным спуском с непрерывным временем[6]:

$$\alpha = K^{-1}(e^{-\eta K} - E)(-Y)$$

Выбран 2й подход из-за меньшей чувствительности к выбору параметра η.

[5] Craig Saunders, Alexander Gammerman и Volodya Vovk. “Ridge regression learning algorithm in dual variables”. B: (1998).

[6] Alnur Ali, J Zico Kolter и Ryan J Tibshirani. “A continuous-time view of early stopping for least squares regression”. B: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, с. 1370—1378.

Задача 2: снижаем сложность оптимизации двойственных переменных

- Применяем метод градиентного бустинга[7].
- Уже есть регрессор $F(z)$ с предыдущего шага бустинга.
- Разобьём X на подмножества: $X^{(1)} \sqcup X^{(2)} \sqcup \ldots \sqcup X^{(l)}$, называемые **батчами**.
- Обучим на каждом из подмножеств линейную модель:

$$\alpha_j = K(X^{(j)})^{-1}(Y^{(j)} - F(X^{(j)}))$$

- Усредним параметры: $\alpha = \frac{1}{l}\sum_j \alpha_j$
- Результатом шага будет уточнённый регрессор:

$$F_{updated}(z) = F(z) + K(z, X)\alpha$$

[7] Jerome H Friedman. “Stochastic gradient boosting”. В: *Computational statistics & data analysis* 38.4 (2002), с. 367—378.

Результаты

Точность классификации на CIFAR-10		
Ядро [8]	Бустинг	Kernel Ridge [9]
Myrtle10 Gaussian Kernel	88.0	88.2
Myrtle10 Gaussian Kernel + Flips	**89.5**	**89.8**
Сложность обучения	$\mathcal{O}(steps\ \mathbf{NM^2})$	$\mathcal{O}(N^3)$

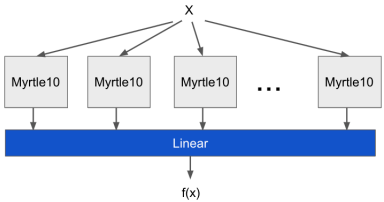
Таблица: Точность классификации (accuracy) на CIFAR-10.

[8] Используются аналитические ядра из материалов статьи (Vaishaal Shankar и др. “Neural kernels without tangents”. В: *International Conference on Machine Learning*. PMLR. 2020, с. 8614—8623)

[9] Колонка Kernel Ridge взята из результатов же статьи.

Задача 3: приближение аналитического ядра

- Построим огромную [10] нейросетевую модель, как конкатенацию маленьких моделей.
- Нейронную сеть только инициализируем, но **не обучаем**.
- В качестве оценки ядра используем скалярные произведения выходов последнего слоя[11].



[10] размер выхода последнего слоя достигает 800 тыс. значений.

[11] Jaehoon Lee и др. “Wide neural networks of any depth evolve as linear models under gradient descent”. В: *Advances in neural information processing systems*. 2019, с. 8570—8581.

Архитектуры моделей

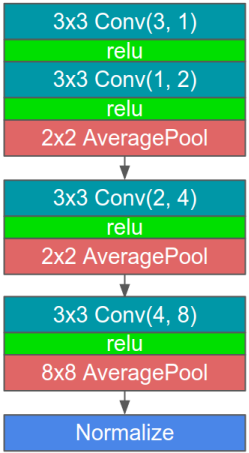


Рис.: Архитектура Myrtle5

Архитектуры моделей

Рис.: Архитектуры Myrtle7 и Myrtle10 соответственно

Результаты на подмножествах CIFAR-10

Точность классификации на подмножествах размера 1280		
Ядро	Оценка ядра	Аналитическое ядро [12]
Myrtle5	$64.2 \pm .3$	$61.9 \pm .7$
Myrtle7	$65.3 \pm .8$	-
Myrtle10	$\mathbf{66.6} \pm \mathbf{.8}$	$64.4 \pm .5$

Таблица: Точность классификации (accuracy) на случайных подмножествах CIFAR-10, состоящих 1280 объектов.

[12] Колонка аналитическое ядро взята из статьи "Neural Kernels Without Tangents". (Vaishaal Shankar и др. "Neural kernels without tangents". В: *International Conference on Machine Learning*. PMLR. 2020, с. 8614—8623)

Результаты на CIFAR-10

Точность классификации на CIFAR-10			
Ядро	Широкая сеть	Оценка ядра	Аналитическое ядро [13]
Myrtle7	-	85.1	**86.6**
Myrtle10	$\leq$ 80. [14]	86.1	**87.5**
Время	-	$\leq$ **9 ч**	$\approx$ 200 ч

Таблица: Точность классификации (accuracy) на CIFAR-10.

Замечание: сравниваемся на ядре, заданном $relu$, а не на гауссовском ядре.

[13] Колонка аналитическое ядро взята статьи (Vaishaal Shankar и др. “Neural kernels without tangents”. В: *International Conference on Machine Learning*. PMLR. 2020, с. 8614—8623)

[14] https://twitter.com/Vaishaal/status/1248293486228459520

Задача4: Сравнить метод с нейронными сетями

Использованы следующие наборы данных:

- **Tiny ImageNet**[15]: 110 тыс. картинок размера 64×64 разбитые на 200 классов. Разделён тренировочную и валидационную выборки в отношении 10:1.
- **Imagenet-R**[16]: 30000 картинок из соревнования ImageNet разбитые на 200 классов.
  Предобработки: каждая картинка отмасштабирована до 224×224, случайное разбиение на тренировочную и тестовую выборки в отношении 9:1.

[15] Jiayu Wu, Qixiang Zhang и Guoxi Xu. *Tiny imagenet challenge*. 2017.

[16] Dan Hendrycks и др. “The many faces of robustness: A critical analysis of out-of-distribution generalization”. В: *arXiv preprint arXiv:2006.16241* (2020).

Результаты на Tiny ImageNet и Imagenet-R

Точность классификации на Tiny ImageNet		
	ядро Myrtle11	ResNet18
без аугментаций	**40.0**	31.8
с аугментацией [17]	**43.4**	36.5

Точность классификации на Imagenet-R		
	ядро Myrtle15	ResNet18
без аугментаций	**24.4**	16.0
с аугментацией [18]	**22.7**	22.1

Таблица: Точности классификации на Tiny ImageNet и Imagenet-R

[17] аугментации: RandomCrop(56), RandomHorizontalFlip

[18] аугментации: RandomCrop(196)

Выводы

- Разработанный ядерный метод хорошо масштабируется по
  - размеру изображений
  - размеру обучающей выборки
- Сложность оптимизации снижена за счёт:
  - батчинга
  - оценки ядра случайными признаками
- Метод превосходит нейросетевой классификатор ResNet18 при обучении с нуля и использовании скудных аугментаций на Tiny-Imagenet.

Список литературы I

- Alnur Ali, J Zico Kolter и Ryan J Tibshirani. “A continuous-time view of early stopping for least squares regression”. В: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, с. 1370—1378.
- Jerome H Friedman. “Stochastic gradient boosting”. В: *Computational statistics & data analysis* 38.4 (2002), с. 367—378.
- Dan Hendrycks и др. “The many faces of robustness: A critical analysis of out-of-distribution generalization”. В: *arXiv preprint arXiv:2006.16241* (2020).
- Arthur Jacot, Franck Gabriel и Clément Hongler. “Neural tangent kernel: Convergence and generalization in neural networks”. В: *Advances in neural information processing systems*. 2018, с. 8571—8580.

Список литературы II

- Jaehoon Lee и др. “Wide neural networks of any depth evolve as linear models under gradient descent”. В: *Advances in neural information processing systems*. 2019, с. 8570—8581.
- Craig Saunders, Alexander Gammerman и Volodya Vovk. “Ridge regression learning algorithm in dual variables”. В: (1998).
- Vaishaal Shankar и др. “Neural kernels without tangents”. В: *International Conference on Machine Learning*. PMLR. 2020, с. 8614—8623.
- Jiayu Wu, Qixiang Zhang и Guoxi Xu. *Tiny imagenet challenge*. 2017.

Решение задачи 1: Обобщим градиентный спуск

Обучаем линейную модель f с параметром θ на объектах X и ответах Y.
Градиентный спуск с функций потерь $L(F) = \frac{1}{2}||F - Y||^2$:

$$f_i(X) = \varphi(X)\theta_i$$

$$\theta_{i+1} = \theta_i - \eta \frac{dL(f_i(X))}{d\theta}$$

Это приближённое решение дифференциального уравнения[19]:

$$f_t(X) = \varphi(X)\theta_t$$

$$\dot{\theta}_t = -\eta \frac{dL(f_t(X))}{d\theta}$$

[19] Jaehoon Lee и др. “Wide neural networks of any depth evolve as linear models under gradient descent”. В: *Advances in neural information processing systems*. 2019.

Решение дифференциального уравнения

Введём двойственные переменные α_t:

$$\alpha_t = K^{-1}(e^{-\eta K t} - E)(-Y)$$

20

Точное решение дифференциального уравнения:

$$f_t(Z) = K(Z, X)\alpha_t$$

$$\theta_t = \varphi^T(X)\alpha_t$$

[20] в дипломной работе показано, что множитель $K^{-1}(e^{-\eta K t} - E)$ можно вычислить для любой матрицы K

Вычисление $K^{-1}(e^{-\eta Kt} - E)$

Вычислим двойственные переменные

$$\alpha_t = K^{-1}(e^{-\eta Kt} - E)(-Y)$$

Заметим, что:

$$\begin{pmatrix} \cdot & K^{-1}(e^{-\eta Kt} - E) \\ \cdot & \cdot \end{pmatrix} = -\eta t \exp \begin{pmatrix} -\eta tK & E \\ 0 & 0 \end{pmatrix}$$

Теорема верна для **необратимой матрицы** K. Доказательство теоремы предложено в дипломной работе.

Вычисление ZCA

- Для повышения точности данные предобрабатываются, преобразованием ZCA.
- Это позволяет сделать исходное пространство более удобным для обучения.
- Время работы реализации преобразования из приложения к статье[21] $\mathcal{O}(P^3 + NP^2)$, где P – число пикселей в изображении, N – число изображений в наборе данных.
- Прошлая реализация этой обработки не позволяла работать с большими изображениями.

[21] Vaishaal Shankar и др. “Neural kernels without tangents”. В: *International Conference on Machine Learning*. PMLR. 2020, с. 8614—8623.

Прошлая реализация ZCA

- Пусть X - матрица объектов размера $N \times P$.
- Тогда результат преобразования $zca(X)$ можно вычислить следующим образом:

$$C = X^T X$$

(матрица ковариаций)

$$C = VD^2V^T$$

(её сингулярное разложение)

$$zca(X) = XV\frac{\sqrt{N-1}}{D+\epsilon}V^T$$

Решение задачи 1: Обобщим градиентный спуск

Обучаем линейную модель f с параметром θ на объектах X и ответах Y.
Градиентный спуск с функций потерь $L(F) = \frac{1}{2}||F - Y||^2$:

$$f_i(X) = \varphi(X)\theta_i$$

$$\theta_{i+1} = \theta_i - \eta \frac{dL(f_i(X))}{d\theta}$$

Это приближённое решение дифференциального уравнения[22]:

$$f_t(X) = \varphi(X)\theta_t$$

$$\dot{\theta}_t = -\eta \frac{dL(f_t(X))}{d\theta}$$

[22] Jaehoon Lee и др. “Wide neural networks of any depth evolve as linear models under gradient descent”. В: *Advances in neural information processing systems*. 2019

Решение дифференциального уравнения

Введём двойственные переменные α_t:

$$\alpha_t = K^{-1}(e^{-\eta K t} - E)(-Y)$$

23

Точное решение дифференциального уравнения:

$$f_t(Z) = K(Z, X)\alpha_t$$

$$\theta_t = \varphi^T(X)\alpha_t$$

[23] в дипломной работе показано, что множитель $K^{-1}(e^{-\eta K t} - E)$ можно вычислить для любой матрицы K

Ускоренная реализация ZCA

- Пусть X - матрица объектов размера $N \times P$.
- $zca(X)$ можно вычислить для больших P следующим образом:

$$X = UDV^T$$

(сингулярное разложение X)

$$zca(X) = XV\frac{\sqrt{N-1}}{D+\epsilon}V^T$$

- С помощью алгоритма *randomized_svd* сложность вычисления падает до $\mathcal{O}(NPlog(k) + (N+P)k^2)$, если вычисляем только k наибольших собственных значений.