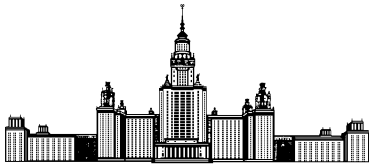
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

Обзорная статья по теме «Kernel Tricks»

Выполнил:

студент 3 курса 317 группы

Борисов Алексей Антонович

Москва, 2021

Содержание

1 Введение

Зачастую теория и алгоритмы машинного обучения очень хорошо изучены для линейного случая. Но для работы с реальными данными часто приходиться использовать нелинейные методы для обнаружения зависимостей, которые помогут успешно решить поставленную задачу. Применение ядерных методов позволяет использовать нелинейные зависимости в изначально казалось бы линейных методах.

Функция $K : X \times X \to \mathbb{R}$ — ядро, если $K(x, x') = \langle \phi(x), \phi(x') \rangle$ при некотором $\phi : X \to H$, где H - гильбертово пространство, которое иногда называют спрямляющим пространством. Таким образом, ядру соответствует скалярное произведение в некотором гильбертовом пространстве (обычно большей размерности чем исходное пространство признаков). Если прогноз зависит от объектов только через попарные скалярные произведения, то можно произвести замену $\langle x, x' \rangle \to K(x, x')$. Это обобщение называется *kernel trick*.[4]

Мотивация применения ядер:

- Исходное пространство X недостаточно, чтобы разделить классы, используя линейные методы, в отличиии от спрямляющего пространства H, соответствующего $\phi(x)$.
- $\langle \phi(x), \phi(x') \rangle$ напрямую вычисляются долго, а $K(x, x')$ быстро.
- $\phi(x)$ может соответствовать переходу в бесконечномерное пространство, например для RBF-ядра.

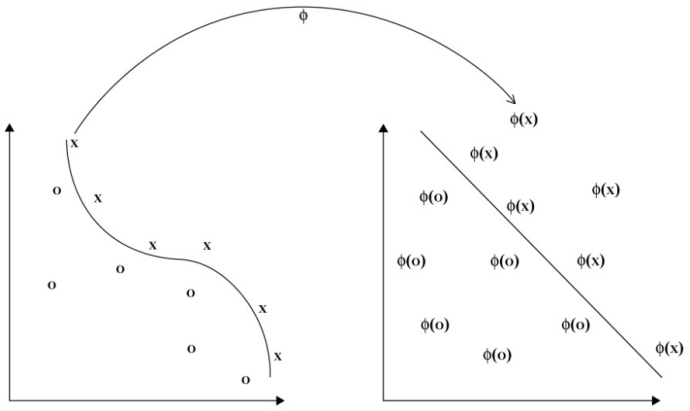


Рис. 1: Отображение объектов в новое пространство.

Приведем несколько популярных примеров ядер:

1. полиномиальное с мономами степени d:
   $K(x, x') = \langle x, x' \rangle^d$
2. полиномиальное с мономами степени $\leq d$:
   $K(x, x') = \big(\langle x, x' \rangle + 1\big)^d$
3. нейросеть с сигмоидными функциями активации:
   $K(x, x') = \mathrm{th}\big(k_1 \langle x, x' \rangle - k_0\big), \quad k_0, k_1 \geq 0$
4. сеть радиальных базисных функций (RBF ядро):
   $K(x, x') = \exp\big(-\gamma \|x - x'\|^2\big)$

2 Использование

Обобщение с помощью ядер допускают многие классические алгоритмы машинного обучения: классификатор опорных векторов, регрессия опорных векторов, гребневая регрессия, метод K-ближайших соседей, метод K-средних, метод главных компонент и другие. Рассмотрим некоторые из них более подробно.

2.1 Метод опорных векторов

Самым известным методом, использующим обобщение с помощью ядер, является метод опорных векторов. Остановимся на классификаторе опорных векторов. Ниже приведена постановка задачи условной минимизации для нахождения оптимальной разделяющей гиперплоскости в линейно неразделимом случае:

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{\ell} \xi_i \to \min\limits_{w, w_0, \xi}; \\ \xi_i \geq 1 - M_i(w, w_0), i = 1, \ldots, \ell; \\ \xi \geq 0, i = 1, \ldots, \ell. \end{cases}$$

где $M_i(w, w_0) = \big(\langle x_i, w \rangle - w_0\big) y_i$ - отступ (margin) объекта x_i.

Эквивалентная задача безусловной минимизации:

$$C\sum_{i=1}^{\ell} \big(1 - M_i(w, w_0)\big)_+ + \frac{1}{2}\|w\|^2 \to \min_{w, w_0}.$$

Используя условия Каруша-Куна-Таккера можно перейти к двойственной задаче:

$$\begin{cases} -\mathscr{L}(\lambda) = -\sum\limits_{i=1}^{\ell} \lambda_i + \frac{1}{2} \sum\limits_{i=1}^{\ell} \sum\limits_{j=1}^{\ell} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle \to \min\limits_{\lambda}; \\ 0 \leq \lambda_i \leq C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0. \end{cases}$$

Решение прямой задачи выражается через решение двойственной:

$$\begin{cases} w = \sum\limits_{i=1}^{\ell} \lambda_i y_i x_i; \\ w_0 = \langle w, x_i \rangle - y_i, \quad \text{для любого } i : \lambda_i > 0, M_i = 1. \end{cases}$$

Линейный классификатор тогда можно записать в виде:

$$a(x) = \operatorname{sign}\Big(\sum_{i=1}^{\ell} \lambda_i y_i \langle x, x_i \rangle - w_0 \Big).$$

Как мы видим, решение зависит от объектов только через скалярные произведения и идея kernel trick состоит в замене $\langle x, x' \rangle$ нелинейной функцией ядра $K(x, x')$. Тогда классификатор примет вид:

$$a(x) = \operatorname{sign}\Big(\sum_{i=1}^{\ell} \lambda_i y_i K(x, x_i) - w_0 \Big).$$

Гиперплоскость в спрямляющем пространстве соответствует нелинейной разделяющей поверхности в исходном пространтсве признаков. Примеры с различными ядрами $K(x, x')$:

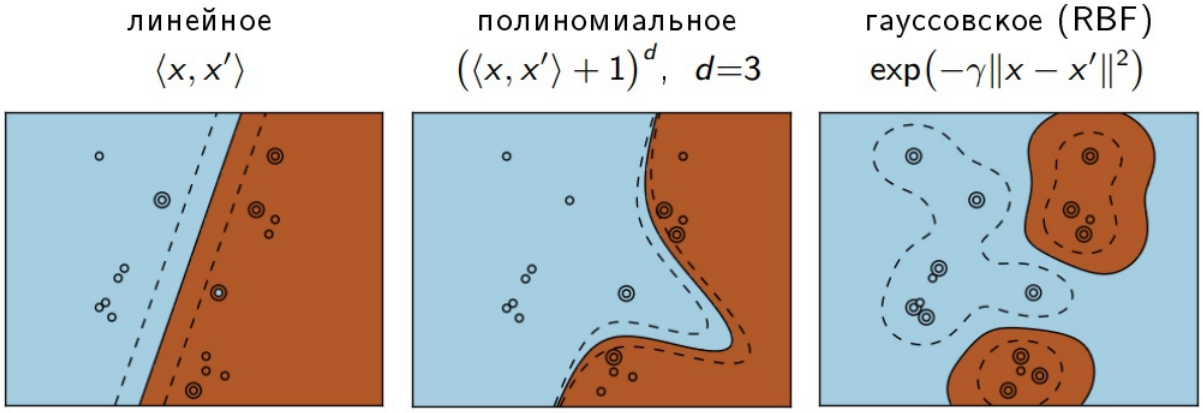


Рис. 2: Разделяющие поверхности с различными ядрами

2.2 Метрические методы

Во многих метрических методах можно применить kernel trick, используя следующие рассуждения.

Евклидово расстояние в исходном пространстве X:

$$\rho(x,z)^2 = \langle x-z, x-z\rangle$$

Тогда в спрямляющем пространстве H:

$$\begin{aligned}\rho(\phi(x),\phi(z))^2 &= \langle \phi(x)-\phi(z), \phi(x)-\phi(z)\rangle \\ &= \langle \phi(x),\phi(x)\rangle + \langle \phi(z),\phi(z)\rangle - 2\langle \phi(x),\phi(z)\rangle \\ &= K(x,x) + K(z,z) - 2K(x,z)\end{aligned}$$

Такое представление позволяет использовать обобщение с помощью ядер для методов K-ближайших соседей, K-средних[6] и других метрических методов.

2.3 Метод главных компонент

Теперь рассмотрим метод главных компонент — очень популярный метод снижения размерности. Пусть у нас имеется датасет $\{x_i\}, i = 1, 2, \dots N$, где x_i представляет собой D-мерный вектор и наша цель спроецировать данные на M-мерное подпространство, где $M < D$. Запишем для такой задачи решение классическим методом главных компонент. Пусть S_x ковариационная матрица $\{x_i\}$, $S_x = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$. $S_x u_k = \lambda_k u_k$, где u_k - собственные векторы S_x. Тогда x_i можно представить в виде:

$$x_i = \sum_{k=1}^{D} (x_i^T u_k) u_k$$

А проекция x_i на M-мерное подпространство будет иметь вид:

$$\widetilde{x_i} = \sum_{k=1}^{M} (x_i^T u_k) u_k$$

где u_k - собственный вектор S_x соответствующий k-ому собственному значению при нумерации от бóльших к меньшим.

Данный метод также можно обобщить с использованием ядер. Пусть $\phi(x_i)$ представление объекта x_i в новом пространстве признаков, причем среднее новых признаковых представлений является нулевым.

$$\frac{1}{N}\sum_{i=1}^{N}\phi(x_i) = 0$$

Тогда ковариационная матрица размера $M \times M$ (M — размерность нового признакового пространства):

$$C = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T$$

Запишем выражение для собственных значений и векторов матрицы C:

$$Cv_k = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)(\phi(x_i)^T v_k) = \lambda_k v_k$$

Собственный вектор матрицы C может быть записан в виде: $v_k = \sum_{i=1}^{N} a_{ki}\phi(x_i)$. Подставляя в предыдущее равенство получим:

$$\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T\sum_{j=1}^{N}a_{kj}\phi(x_j) = \lambda_k\sum_{i=1}^{N}a_{ki}\phi(x_i)$$

Теперь введем функцию ядра $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ Домножив обе части равенства на $\phi(x_l)$, заметим, что выражение зависит от объектов только через скалярные произведения $\phi(x_i)^T\phi(x_j)$, что позволяет применить kernel trick.

$$\frac{1}{N}\sum_{i=1}^{N}K(x_l, x_i)\sum_{j=1}^{N}a_{kj}K(x_i, x_j) = \lambda_k\sum_{i=1}^{N}a_{ki}K(x_l, x_i)$$

В матричной записи получим: $K^2 a_k = \lambda_k N K a_k$, где $K_{i,j} = K(x_i, x_j)$. a_k может быть найдена из $Ka_k = \lambda_k N a_k$, и главные компоненты для объекта x имеют вид:

$$y_k(x) = \phi(x)^T v_k = \sum_{i=1}^{N}a_{ki}K(x, x_i)$$

и тогда проекция $\phi(x)$ на m-мерное подпространство имеет вид:

$$P_m\phi(x) = \sum_{k=1}^{m}y_k(x)v_k$$

Случай, когда среднее новых признаковых представлений объектов не равно нулю, а также более подробные выкладки можно найти в соответствующей статье [9].

2.4 Multi-layer kernel machines

В [2] Youngmin Cho представил многослойную ядерную машину (Multi-layer Kernel Machine, MKM). В MKM нелинейные трансформации производятся ядерными функциями, которые образуют слои. На рисунке 3 представлена архитектура L-слойной MKM.[7] Каждый слой включает в себя извлечение признаков с использованием ядерного метода главных компонент и последующий отбор признаков на основе их взаимной информации.

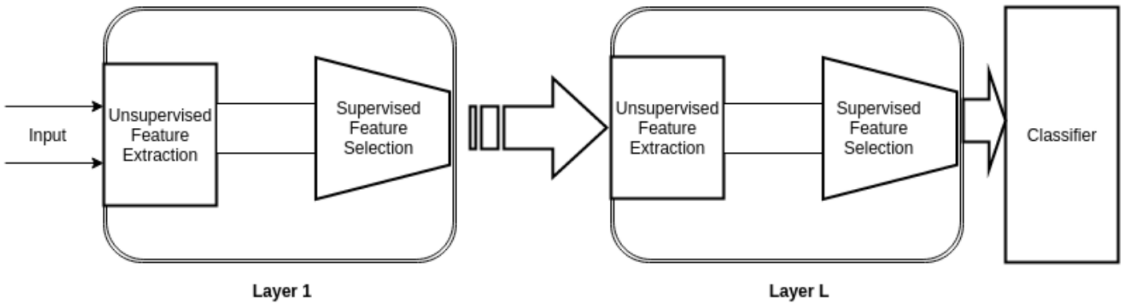


Рис. 3: Архитектура MKM с L слоями.

В статье определяют понятие L-слойной ядерной функции:

$$K^{(L)}(x,y) = \langle \underbrace{\phi(\phi(\dots\phi(x)))}_{\text{L раз}}, \underbrace{\phi(\phi(\dots\phi(y)))}_{\text{L раз}} \rangle$$

Авторы используют арккосинусное ядро, больше о котором можно прочитать в [7]. Интуиция использования многослойных ядер заключается в том, что если одно ядро способно имитировать однослойную нейронную сеть, то последовательное использование нескольких ядер позволит имитировать многослойную нейронную сеть.

Также предлагается использовать подход Multiple Kernel Learning (MKL), который заключается в обучении ядра в виде выпуклой комбинации нескольких заранее определенных ядер с целью обучить ядро, которое лучше подходит в конкретной задаче.

$$K_{conv} = \left\{ K(\cdot,\cdot) = \sum_{t=1}^{m} \mu_t K_t(\cdot,\cdot) : \sum_{t=1}^{m} \mu_t = 1, \mu_t \geq 0 \right\}$$

где K_t - ядра из заранее определенного набора.

2.5 Ядерные методы и нейронные сети

Как уже стало понятно из предыдущего раздела, развитие нейронных сетей повлияло и на использование ядерных методов. Исторически нейронные сети и ядерные методы развивались параллельно, но сейчас существуют попытки объединить эти два подхода. Например, в статье [8] подробно обсуждаются отличия архитектуры нейронных сетей и моделей, использующих ядра.

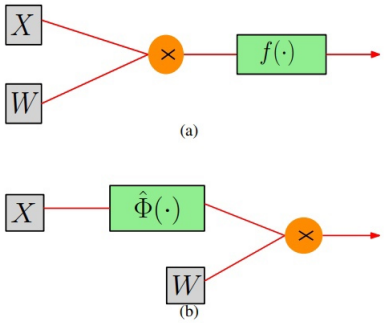


Рис. 4: (a) Один блок нейросетевой модели. (b) Один блок модели на основе ядер.

После чего рассказывается о гибридных моделях, комбинирующих подходы из нейронных сетей и из ядерных методов. В статье они называются deep hybrid neural-kernel networks. Также в [8] авторы представляют собственные архитектуры блоков для гибридных моделей. Всего они рассматривают 3 варианта: deep average neural-kernel network, deep maxout neural-kernel network, а также deep convolutional neural-kernel network. Более подробную информацию о которых можно найти в соответствующей статье.

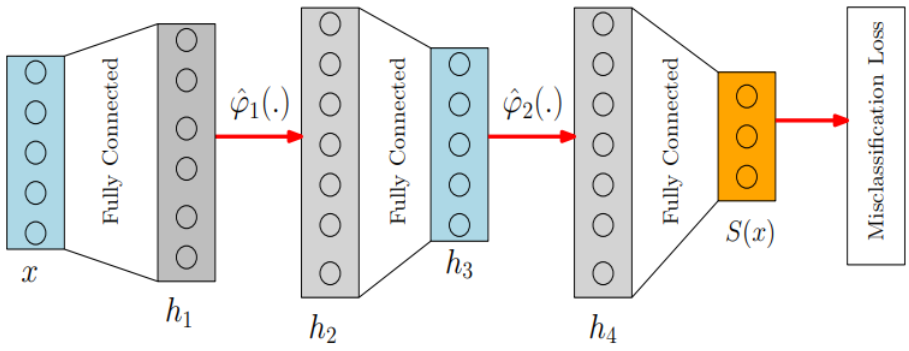


Рис. 5: Архитектура deep hybrid neural-kernel network

2.6 Ядра для объяснения работы трансформеров

Несмотря на повсеместное использование в обработке естественного языка, механизм работы глубоких нейронных сетей, построенных на основе механизма внимания (attention), остается до конца непонятен. В статье [11] авторы рассматривают трансформеры как kernel machines и особое внимание уделяют механизму внимания.

Напомним некоторую информацию о работе механизма внимания. В общей формулировке имеются два множества элементов: исходные элементы $\{s_1, s_2, \ldots, s_S\}, s_j \in \mathbb{R}^{d_s}$ и множество целевых элементов $\{t_1, t_2, \ldots, t_T\}, t_i \in \mathbb{R}^{d_t}$. Целью является создание для каждого элемента t_i специального эмбеддинга последовательности исходных элементов $\{s_j\}_{j=1}^S$, учитывающего специфику элемента t_i.

$$a_{ij} = \frac{(W^Q t_i)^T (W^K s_j)}{\sqrt{d}} \quad \alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{j=1}^S \exp(a_{ij})} \quad t_i' = \sum_{j=1}^S \alpha_{ij} W^V s_j$$

где $W^V, W^K \in \mathbb{R}^{d_s \times d}$, и $W^Q \in \mathbb{R}^{d_t \times d}$ обучаемые матрицы весов, которые обычно называют «value», «key» и «query» весовые матрицы. Для того чтобы представить механизм внимания в виде ядра, авторы вводят новые представления $\Phi_{\mathcal{X}}(t)$ и $\Phi_{\mathcal{Y}}(s)$ для элементов, а также банаховы пространства $\mathcal{B}_{\mathcal{X}}$ и $\mathcal{B}_{\mathcal{Y}}$. После создания необходимых конструкций авторы показывают, что теперь механизм внимания можно представить в виде функции ядра, которую они назвали «exponentiated query-key kernel»:

$$K(t,s) = \langle \Phi_{\mathcal{X}}(t), \Phi_{\mathcal{Y}}(s) \rangle_{\mathcal{F}_{\mathcal{X}} \times \mathcal{F}_{\mathcal{Y}}} = \langle f_{\Phi_{\mathcal{X}}(t)}, g_{\Phi_{\mathcal{Y}}(s)} \rangle_{\mathcal{B}_{\mathcal{X}} \times \mathcal{B}_{\mathcal{Y}}} = \exp\Big(\frac{(W^Q t)^T (W^K s)}{\sqrt{d}}\Big)$$

После этого много внимания уделено представлению самого трансформера в виде kernel learner, а также использованию других ядер. Наилучшие результаты в экспериментах показало ядро, которое и используется в оригинальных моделях трансформера. По этой причине авторы задаются вопросом: всегда ли предпочительно использовать стандартное ядро, или есть приложения, в которых другие ядра могут дать лучшие результаты. Более подробную информацию о выводе механизма внимания в виде ядра и экспериментах можно найти в соответствующей статье [11].

2.7 Deep kernel machines

Вдохновившись успехом глубоких нейронных сетей, в статье [1] авторы представили свою модель — deep kernel machine. Сперва в статье рассказывается о вероятностном подходе с использованием deep kernel process (DKP). В DKP используется определенное количество промежуточных слоев, каждый из которых содержит ядро, априорное и приближенное апостериорное распределения по этому ядру. В статье отмечается, что deep kernel process обобщают байесовские нейронные сети (BNNs) и глубокие гауссовы процессы (DGPs), что указывает на то, что они параметризуют мощный, гибкий класс методов. Но авторы предлагают использовать небайесовские глубокие ядерные методы, deep kernel machine.

Итоговый алгоритм заключается в использовании итерационного метода для решения непрерывного алгебраического уравнения Рикатти (CARE):

$$0 = A^T G + GA - GUG + Q$$

где G, U и Q положительно определенные матрицы, а A — вещественная матрица.

В следующей части статьи авторы занимаются оптимизацией алгоритма. В результате авторы предоставили оптимизатор для deep kernel machine, который в корне отличается от стандартных градиентных методов, поскольку включает в себя решение CARE с использованием подходов из теории управления. На практике представленная решающая программа способна показывать хорошие результаты и при этом работает значительно быстрее чем эквивалентные методы, основанные на градиентном спуске.

2.8 Ядерные методы для улучшения качества речи

Еще одно интересное приложение ядерных методов представлено в статье [5]. В ней авторы используют ядерные методы для улучшения качества речи и получают результаты лучше чем при использовании глубоких нейронных сетей. При этом время обучения их метода меньше чем при использовании нейронных сетей. В статье решают задачу ядерной регрессии, используя специальную негладкую ядерную функцию (экспоненциаьное степенное ядро). Для обучения применяется эффективный итерационный метод EigenPro.

$$K_{\gamma,\sigma}(x,z) = \exp\Big(-\frac{\|x-z\|^{\gamma}}{\sigma}\Big)$$

Как рассказывают авторы, в силу простоты метода некоторые гиперпараметры модели, например, значение σ в функции ядра можно находить с помощью линейного поиска, используя подвыборки обучающих данных. В статье отмечается, что для многих видов шума оптимальное значение параметра γ получается меньше единицы, чему соответствует негладкая функция ядра.

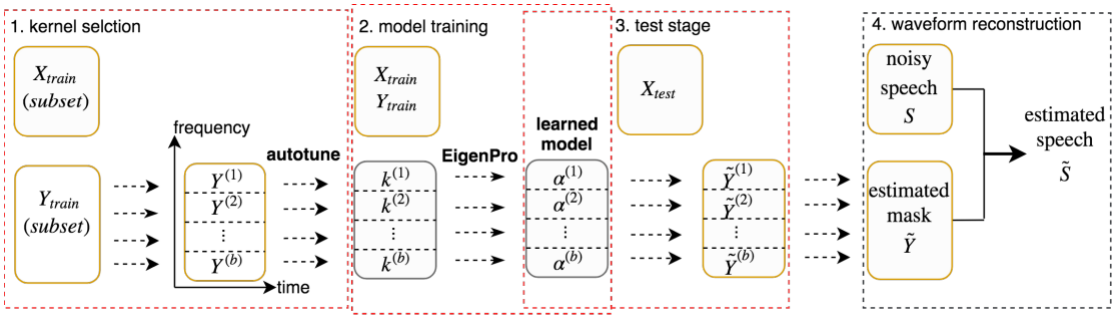


Рис. 6: Архитектура модели, использующей ядра, для улучшения качества речи

3 Выводы

Изначально kernel trick появился как возможность оптимизировать вычисления и перейти в пространства большей размерности для использования в них хорошо изученных линейных методов, что мы могли видеть на примере метода опорных векторов и метода главных компонент. Но в последнее время с помощью ядер пробуют объяснить работу нейронных сетей, так как большинство современных state-of-the-art архитектур представляет собой «черный ящик», с объяснением работы которого возникают определенные трудности.

Список литературы

[1] Laurence Aitchison. Deep kernel machines and fast solvers for deep kernel machines, 2021.

[2] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[3] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nyström method, and use of kernels in machine learning: Tutorial and survey, 2021.

[4] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3), Jun 2008.

[5] Like Hui, Siyuan Ma, and Mikhail Belkin. Kernel machines beat deep neural networks on mask-based single-channel speech enhancement, 2018.

[6] Mieczysław A. Kłopotek. Validity of clusters produced by kernel-k-means with kernel-trick, 2018.

[7] Akhil Meethal, Asharaf S, and Sumitra S. Unsupervised mkl in multi-layer kernel machines, 2021.

[8] Siamak Mehrkanoon. Deep neural-kernel machines, 2020.

[9] Quan Wang. Kernel principal component analysis and its applications in face recognition and active shape models, 2014.

[10] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning, 2015.

[11] Matthew A. Wright and Joseph E. Gonzalez. Transformers are deep infinite-dimensional non-mercer binary kernel machines, 2021.