Лекция 14
Ядра в машинном обучении

Е. А. Соколов
ФКН ВШЭ

1 февраля 2019 г.

Ядра позволяют превращать линейные методы машинного обучения в нелинейные за счёт подмены признакового пространства. При этом, поскольку подмена производится через скалярное произведение, сложность методов не повышается. Мы уже знаем, как конструируются ядра, а также изучили несколько их распространённых примеров — например, полиномиальные и гауссовы ядра. Теперь мы обсудим вычислительные трудности, связанные с ядровыми методами, и разберём методы их устранения с помощью рандомизации. Далее мы обсудим возможность применения ядер к сложным объектам на примере строковых данных. Наконец, мы разберём ещё одно применение ядер — а именно, в методе главных компонент.

1 Аппроксимация спрямляющего пространства

Все ядровые методы используют матрицу Грама $G = XX^T$ вместо матрицы «объекты-признаки» X. Это позволяет сохранять сложность методов при сколь угодно большой размерности спрямляющего пространства, но работа с матрицей Грама для больших выборок может стать затруднительной. Так, уже при выборках размером в сотни тысяч объектов хранение этой матрицы потребует большого количества памяти, а обращение станет трудоёмкой задачей, поскольку требует $O(\ell^3)$ операций.

Решением данной проблемы может быть построение в явном виде такого преобразования $\tilde{\varphi}(x)$, которое переводит объекты в пространство не очень большой размерности, и в котором можно напрямую обучать любые модели. Мы разберём метод случайных признаков Фурье (иногда также называется Random Kitchen Sinks) [2], который обладает свойством аппроксимации скалярного произведения:

$$\langle \tilde{\varphi}(x), \tilde{\varphi}(z) \rangle \approx K(x, z).$$

Из комплексного анализа известно, что любое непрерывное ядро вида $K(x, z) = K(x - z)$ является преобразованием Фурье некоторого вероятностного распределения (теорема Бохнера):

$$K(x - z) = \int_{\mathbb{R}^d} p(w) e^{iw^T(x-z)} dw.$$

Преобразуем интеграл:

$$\int_{\mathbb{R}^d} p(w)e^{iw^T(x-z)}dw = \int_{\mathbb{R}^d} p(w)\cos(w^T(x-z))dw + i\int_{\mathbb{R}^d} p(w)\sin(w^T(x-z))dw = \\ = \int_{\mathbb{R}^d} p(w)\cos(w^T(x-z))dw.$$

Поскольку значение ядра $K(x-z)$ всегда вещественное, то и в правой части мнимая часть равна нулю — а значит, остаётся лишь интеграл от косинуса $\cos w^T(x-z)$. Мы можем приблизить данный интеграл методом Монте-Карло:

$$\int_{\mathbb{R}^d} p(w)\cos w^T(x-z)dw \approx \frac{1}{n}\sum_{j=1}^{n}\cos w_j^T(x-z),$$

где векторы $w_1, \ldots, w_n$ генерируются из распределения $p(w)$. Используя эти векторы, мы можем сформировать аппроксимацию преобразования $\varphi(x)$:

$$\tilde{\varphi}(x) = \frac{1}{\sqrt{n}}(\cos(w_1^T x), \ldots, \cos(w_n^T x), \sin(w_1^T x), \ldots, \sin(w_n^T x)).$$

Действительно, в этом случае скалярное произведение новых признаков будет иметь вид

$$\tilde{K}(x,z) = \langle\tilde{\varphi}(x), \tilde{\varphi}(z)\rangle = \frac{1}{n}\sum_{j=1}^{n}\left(\cos(w_j^T x)\cos(w_j^T z) + \sin(w_j^T x)\sin(w_j^T z)\right) \\ = \frac{1}{n}\sum_{j=1}^{n}\cos w_j^T(x-z).$$

Данная оценка является несмещённой для $K(x,z)$ в силу свойств метода Монте-Карло. Более того, с помощью неравенств концентрации меры можно показать, что дисперсия данной оценки достаточно низкая. Например, для гауссова ядра будет иметь место неравенство

$$\mathbb{P}\left[\sup_{x,z}|\tilde{K}(x,z) - K(x,z)| \geqslant \varepsilon\right] \leqslant 2^8(2d\sigma^2/\varepsilon)^2\exp(-d\varepsilon^2/4(d+2)).$$

Разумеется, найти распределение $p(w)$ можно не для всех ядер $K(x-z)$. Как правило, данный метод используется для гауссовых ядер $\exp(\|x-z\|^2/2\sigma^2)$ — для них распределение $p(w)$ будет нормальным с нулевым матожиданием и дисперсией σ^2.

2 All-subsequences kernel

Рассмотрим ядро, часто используемое при работе с текстами. Введём некоторые понятия и обозначения:

- Σ — алфавит, некоторое множество элементов, называемых *символами*;
- Σ^* — множество всех возможных последовательностей (называемых *строками*; включая пустую строку ε) над алфавитом Σ;

- $|s|$ — длина строки s;
- $\overline{s_1 s_2 \dots s_k}$ — конкатенация символов или строк $s_1, s_2, \dots, s_k$;
- $s(i)$ — подпоследовательность символов строки s на позициях $i = (i_1, \dots, i_k), 1 \le i_1 < \dots < i_k \le |s|$, т.е. строка $\overline{s_{i_1} \dots s_{i_k}}$;
- $s[a:b] = \begin{cases} s((a, a+1, \dots, b)), a \le b, \\ \varepsilon, a > b. \end{cases}$

Для произвольной строки над алфавитом Σ рассмотрим следующее отображение в спрямляющее пространство:

$$(\varphi(s))_u = |\{i : s(i) = u\}|, u \in \Sigma^*,$$

т.е. $(\varphi(s))_u$ — количество вхождений строки u в строку s в качестве её подпоследовательности. Соответствующее ядро задаётся следующим образом:

$$K(s,t) = \langle \varphi(s), \varphi(t) \rangle = \sum_{u \in \Sigma^*} (\varphi(s))_u (\varphi(t))_u.$$

Тем не менее, вычисление ядра путём формирования признаковых описаний объектов слишком трудозатратно, даже если учитывать лишь подпоследовательности, действительно входящие в строку в исходном пространстве. В частности, для подпоследовательностей длины k количество ненулевых компонент признакового описания строки s в спрямляющем пространстве можно оценить как $\min\left(C^k_{|s|}, |\Sigma|^k\right)$.

Опишем более эффективный способ вычисления ядра. Преобразуем вклад $(\varphi(s))_u$ в значение $K(s,t)$:

$$(\varphi(s))_u (\varphi(t))_u = \sum_{i: s(i)=u} 1 \cdot \sum_{j: t(j)=u} 1 = \sum_{(i,j):\, u=s(i)=t(j)} 1.$$

Тогда

$$K(s,t) = \langle \varphi(s), \varphi(t) \rangle = \sum_{u \in \Sigma^*} \sum_{u=s(i)=t(j)} 1 = \sum_{(i,j):\, s(i)=t(j)} 1.$$

Для эффективного вычисления ядра будем использовать рекуррентную формулу — вычислим значение ядра $K(sa,t)$, где a — символ, дописанный в конец рассматриваемой ранее строки:

$$K(sa,t) = \sum_{(i,j):\, sa(i)=t(j)} 1.$$

В этом случае для набора i возможны 2 случая: i целиком содержится в s либо последний элемент i является символом a. Таким образом, имеем:

$$\sum_{(i,j):\overline{sa}(i)=t(j)} 1 = \sum_{(i,j):s(i)=t(j)} 1 + \sum_{u:t=\overline{uav}} \sum_{(i,j):\, s(i)=u(j)} 1.$$

При разбиении суммы мы воспользовались тем фактом, что при наличии совпадающих подпоследовательностей в строках sa и t с участием символа a в первой из них этот символ должен также встречаться на некоторой позиции в строке t.

Описанные преобразования позволяют нам сформулировать реккурентную формулу для вычисления ядра:

$$K(s,\varepsilon) = 1,$$
$$K(sa,t) = K(s,t) + \sum_{k:\, t_k = a} K(s, t[1:k-1]).$$

Верны также и аналогичные симметричные формулы в силу симметричности ядра.

Таким образом, для вычисления значения ядра можно составить таблицу размера $(|s|+1)(|t|+1)$. Обозначим за $DP(i,j)$ значение в позиции (i,j), $i = \overline{0,|s|}$, $j = \overline{0,|t|}$, этой таблицы и будем заполнять таблицу таким образом, чтобы в позиции(i,j) находилось значение $K(s[1:i], t[1:j])$.

	j	0	1	2	$\dots$	j	$\dots$	$\lvert t\rvert$
i		ε	t_1	t_2	$\dots$	t_j	$\dots$	$t_{\lvert t\rvert}$
0	ε	1	1	1	$\dots$	1	$\dots$	1
1	s_1	1	$\ddots$	$\ddots$				$\vdots$
2	s_2	1	$\ddots$	$\ddots$				
$\vdots$	$\vdots$	$\vdots$				$\vdots$		
i	s_i	1				$K(s[1:i], t[1:j])$		$\vdots$
$\vdots$	$\vdots$	$\vdots$					$\ddots$	$\vdots$
$\lvert s\rvert$	$s_{\lvert s\rvert}$	1	$\dots$			$\dots$	$\dots$	$K(s,t)$

При этом согласно реккурентной формуле имеем:

$$DP(1,j) = DP(i,1) = 1, i = \overline{1,|s|+1}, j = \overline{1,|t|+1},$$
$$DP(i,j) = DP(i-1,j) + \sum_{k \le j:\, t_k = s_i} DP(i-1, k-1),$$

поэтому таблицу можно заполнять по строкам сверху вниз слева направо. Можно заметить, для вычисления $DP(i,j)$ требуются значения $DP(i-1,k)$, $k = \overline{0,j-1}$, а потому заполнение позиции (i,j) таблицы требует $O(j)$ операций, откуда следует, что вычисление значения ядра $K(s,t) = DP(|s|,|t|)$ требует $O(|s||t|^2)$ операций.

Заметим, что при заполнении i-ой строки таблицы сумма в реккурентной формуле для $DP(i,j)$ использует один и тот же символ s_i, причём каждая последующая сумма включает в себя предыдущие, а потому они могут быть вычислены динамически заранее для i-ой строки путём прохода по строке t, поиска символов s_i и прибавления соответствующего слагаемого суммы в случае успешного нахождения. Обозначив полученный вектор сумм за P, можем вычислять значение в позиции (i,j) таблицы по следующей формуле:

$$DP(i,j) = DP(i-1,j) + P(j).$$

Отметим, для вычисления значений сумм для i-ой строки таблицы требуется $O(|t|)$ операций, а потому полученный алгоритм вычисления ядра $K(s,t)$ имеет сложность $O(|s||t|)$.

3 Ядровой метод главных компонент

Вспомним, что в методе главных компонент вычисляются собственные векторы $u_1, \ldots, u_d$ ковариационной матрицы $X^T X$, соответствующие наибольшим собственным значениям. После этого новое признаковое описание объекта x вычисляется с помощью его проецирования на данные компоненты:

$(\langle u_j, x\rangle)_{j=1}^d$.

Попробуем теперь воспользоваться методом главных компонент в ядровом пространстве, где объекты описываются векторами $\varphi(x)$. Поскольку зачастую отображение $\varphi(x)$ нельзя выписать в явном виде, сформулируем метод главных компонент в терминах матрицы Грама $K = \Phi\Phi^T$ и ядра $K(x, z)$. Отметим, что напрямую пользоваться ковариационной матрицей $\Phi^T\Phi$ нельзя, поскольку она имеет размер $d \times d$, а число признаков d в спрямляющем пространстве может быть слишком большим; более того, спрямляющее пространство может быть бесконечномерным, и в этом случае ковариационную матрицу получить вообще не получится.

Пусть v_j — собственный вектор матрицы Грама K, соответствующий собственном значению λ_j. Рассмотрим цепочку уравнений:

$$\Phi^T\Phi(\Phi^T v_j) = \Phi^T(\Phi\Phi^T v_j) = \lambda_j \Phi^T v_j,$$

из которой следует, что $\Phi^T v_j$ является собственным вектором ковариационной матрицы $\Phi^T\Phi$, соответствующим собственному значению λ_j. Найдём норму данного вектора:

$$\|\Phi^T v_j\|^2 = v_j^T(\Phi\Phi^T v_j) = \lambda_j v_j^T v_j = \lambda_j,$$

где мы воспользовались нормированностью собственных векторов v_j. Значит, векторы $u_j = \lambda_j^{-1/2}\Phi^T v_j$ будут являться ортонормированной системой собственных векторов ковариационной матрицы.

Преобразуем выражение для u_j:

$$u_j = \lambda_j^{-1/2} \sum_{i=1}^{\ell} (v_j)_i \varphi(x_i) = \sum_{i=1}^{\ell} \alpha_{ji}\varphi(x_i),$$

где $\alpha_{ji} = \lambda_j^{-1/2} v_j$.

Мы выразили главные компоненты через признаковые описания объектов обучающей выборки в ядровом пространстве. Теперь найдём проекции объекта $\varphi(x)$ на эти компоненты:

$$\begin{aligned}\langle u_j, \varphi(x)\rangle &= \left\langle \sum_{i=1}^{\ell} \alpha_{ji}\varphi(x_i), \varphi(x) \right\rangle \\ &= \sum_{i=1}^{\ell} \alpha_{ji}\langle \varphi(x_i), \varphi(x)\rangle \\ &= \sum_{i=1}^{\ell} \alpha_{ji} K(x_i, x).\end{aligned}$$

Итак, мы выразили проекции на главные компоненты через ядро и через собственные векторы матрицы Грама — этого достаточно, чтобы вычислять проекции, не используя напрямую признаковые описания объектов из спрямляющего пространства.

Список литературы

[1] *Drineas, Petros and Mahoney, Michael W.* On the NyströM Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. // Journal of Machine Learning Research, 2005.

[2] *Rahimi, Ali and Recht, Benjamin* Random Features for Large-scale Kernel Machines. // Proceedings of the 20th International Conference on Neural Information Processing Systems, 2007.