大数据算法-2025 春

Lecture 7: 数据降维之 JL 变换

2025.3.20

Lecturer: 丁虎 *Scribe:* 沈俊杰，王运韬

Johnson-Lindenstrauss（JL）变换是一种用于高维数据的降维技术。其基本思想是将高维数据投影到一个低维的欧氏空间，同时保持数据点之间的距离。

1 JL 引理

Theorem 1.1 (JL 引理)**.** 对于任意给定的 $\epsilon \in (0,1)$ 和 $P \subset \mathbb{R}^d, |\mathrm{P}| = n \in \mathbb{N}$，存在一个映射 $f : \mathbb{R}^d \to \mathbb{R}^k$，其中 $k = O(\frac{1}{\epsilon^2}\log n)$，使得对于任意两个向量 $x, y \in P$, 有

$$(1-\epsilon)||x-y||^2 \leq ||f(x)-f(y)||^2 \leq (1+\epsilon)||x-y||^2 \tag{1}$$

这个引理的含义是，对于任意的两个向量 x, y，在映射后的空间中，它们的距离与原空间中的距离保持在 $(1-\epsilon, 1+\epsilon)$ 的范围内。换句话说，f 尽量模仿了刚体变换，因为刚体变换就是保距变换 [4], 即式(1) 在 $\epsilon = 0$ 的情形.

Definition 1.2. JL 变换

构造一个矩阵 $\mathrm{B} \in \mathbb{R}^{k\times d}$

$$f(u) = \mathrm{B} \cdot u \tag{2}$$

如果 f 满足 JL 引理中的条件，那么称 B 为 JL 变换。

很自然的，我们需要考虑如何构造这样的矩阵 B.

2 构造 JL 变换的方法

一种简单的构造方法是，通过高斯分布进行构造：$\mathrm{B} = \frac{1}{\sqrt{k}} \cdot \mathrm{A}$，其中 $a_{ij} \sim N(0,1)$

现在我们需要证明的是，这样构造的 B 是一个 JL 变换。注意到矩阵计算的性质，实际上证明 JL 引理成立仅需要证明对于 $\forall x \in \mathbb{X}$，有：

$$(1-\epsilon)||x||^2 \leq ||\mathrm{B} \cdot x||^2 \leq (1+\epsilon)||x||^2 \tag{3}$$

其中 $\mathbb{X}$ 中的元素集合 $\mathbb{P}$ 元素的点对组合，$|\mathbb{X}| = \binom{n}{2}$ 我们可以考虑证明对某一个 x 作证明，然后通过 union bound 的技术推广到整个集合。

$y = \frac{1}{\sqrt{k}}\mathrm{A} \cdot x$，因为随机矩阵 B 的引入，$y$ 是一个随机变量，自然的，我们应当考虑他的性质，比如 $E[y^T y]$。

Lemma 2.1.

$$E[\|y\|^2] = \|x\|^2$$

Proof.

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}$$

$$\begin{aligned}
E[\|y\|^2] &= \frac{1}{k}E[\sum_{j=1}^{k}(A_j x)^2] \\
&= \frac{1}{k}\sum_{j=1}^{k}E[(A_j x)^2] \\
&= \frac{1}{k}\sum_{j=1}^{k}E[\sum_{i=1}^{d}(A_{ji}x_i)^2] \\
&= \frac{1}{k}\sum_{j=1}^{k}E[\sum_{i/i'}^{d}A_{ji}A_{ji'}x_i x_{i'}] \\
\forall i \neq i' \quad & E[A_{ji}A_{ji'}] = E[A_{ji}]E[A_{ji'}] = 0 \quad \text{(独立性)} \\
\Rightarrow E[(A_j x)^2] &= \sum_{i=1}^{d}E[A_{ji}^2 x_i^2] \\
&= \sum_{i=1}^{d}x_i^2 \\
\therefore E[\|y\|^2] &= \|x\|^2
\end{aligned}$$

□

仅仅期望，是不够的，当然，我们可以通过根据期望相关的不等式得到一些相对粗糙的结论，不在此赘述。我们考虑 Prob $[\|y\|^2 \geq (1+\epsilon)\|x\|^2]$，此即 Jl 引理的右半边形式。

简单化简，Prob 内的公式可化为：

$$\frac{\|Ax\|^2}{\|x\|^2} \geq (1+\epsilon)k$$

考虑左边：

$$\begin{aligned}
\frac{\|Ax\|^2}{\|x\|^2} &= \sum_{j=1}^{k} \frac{(A_j \cdot x)^2}{\|x\|^2} \\
&= \sum_{j=1}^{k} z_j^2 \qquad\qquad (z_j = \frac{A_j \cdot x}{\|x\|} = \sum_{i=1}^{d} A_{ji} \frac{x}{\|x\|})
\end{aligned}$$

注意到 $z_j \sim N(0,1)$，因此我们考虑 Prob $\left[\sum_{j=1}^{k} z_j^2 \geq (1+\epsilon)k\right]$

$$\begin{aligned}
\text{Prob}\left[\sum_{j=1}^{k} z_j^2 \geq (1+\epsilon)k\right] &= \text{Prob}\left[exp(\lambda \sum_{j=1}^{k} z_j^2) \geq exp(\lambda(1+\epsilon)k)\right] \\
&\leq \frac{1}{exp(\lambda(1+\epsilon)k)} E\left[exp(\lambda \sum_{j=1}^{k} z_j^2)\right] \qquad (Markov Inequality) \\
&= \frac{1}{exp(\lambda(1+\epsilon)k)} \prod_{j=1}^{k} E\left[exp(\lambda z_j^2)\right] \\
&= \frac{1}{exp(\lambda(1+\epsilon)k)} \prod_{j=1}^{k} \frac{1}{\sqrt{1-2\lambda}} \qquad (z_j^2 \sim \chi^2(1)\text{矩母函数}) \\
&= \frac{1}{exp(\lambda(1+\epsilon)k)} \frac{1}{(1-2\lambda)^{k/2}} \\
&= ((1+\epsilon)e^{-\epsilon})^{k/2} \qquad (\text{let } \lambda = \frac{\epsilon}{2(1+\epsilon)}) \\
&\leq e^{-\frac{k}{4}(\epsilon^2-\epsilon^3)}
\end{aligned}$$

对于另一边不等式，我们有类似的推导，最终得到：

$$\begin{aligned}
\text{Prob}\left[\|y\|^2 \geq (1+\epsilon)\|x\|^2\right] &\leq e^{-\frac{k}{4}(\epsilon^2-\epsilon^3)} \\
\text{Prob}\left[\|y\|^2 \leq (1-\epsilon)\|x\|^2\right] &\leq e^{-\frac{k}{4}(\epsilon^2+\epsilon^3)} \\
\Rightarrow \text{Prob}\left[\|y\|^2 \in (1\pm\epsilon)\|x\|^2\right] &\geq 1 - 2e^{-\frac{k}{4}(\epsilon^2-\epsilon^3)}
\end{aligned}$$

一共有 $\binom{n}{2} = \Theta(n^2)$ 个点对，我们可以通过 union bound 的技术，若要得到常数概率，我们需要 $k = \Theta(\frac{1}{\epsilon^2}\log n)$ 即可，通过反解下式得到。

$$e^{-\frac{k}{4}(\epsilon^2-\epsilon^3)} = \Theta(\frac{1}{n^2})$$

Remark 2.2. 关于概率的一些结算，请牢记证明中引入的随机变量是 $A = (A_{ij})$，一切随机性由其得到。

Remark 2.3. Key idea:

$$\begin{aligned} \|y\|^2 &= y_1^2 + y_2^2 + \cdots + y_k^2 \\ &= \frac{1}{k}((A_1x)^2 + (A_2x)^2 + \cdots + (A_kx)^2) \qquad (\text{每一项都是对}\|x\|^2\text{的估计}) \\ A_1x &= A_{11}x_1 + A_{12}x_2 + \cdots + A_{1d}x_d \end{aligned}$$

不妨假设x是单位向量，那么$(A_1x)^2$以一定概率接近 1 即可

3 JL 变换 v.s. PCA

Table 1: JL 变换 v.s. PCA

	JL 变换	**PCA**
Runnning time	$\Theta(nd\frac{\log n}{\epsilon^2})$	$\Theta(nd^2)$
Data	Data Oblivious(可应用于流数据，并行)	Data Dependent

Remark 3.1. 这里的 JL 变换时间复杂度中，n 来自于对于 n 个数据点依次进行 JL 变换，但理论上，我们不需要知道所有点就可以进行 JL 变换，但 PCA 做不到这一点，PCA 需要对 n 个数据点一起处理。这也是所谓 Data Oblivious 的意思。

4 其他 JL 变换

基于 Gaussian 的矩阵 A 是稠密的，如果我们想要一个稀疏的随机矩阵，是否有办法呢？答案是肯定的。事实上，有一些别的构造方法，不依赖于 Gaussian。例如：

$$A_{ij} = \begin{cases} +1 & \sim 50\% \\ -1 & \sim 50\% \end{cases} \qquad A_{ij} = \sqrt{3}\begin{cases} +1 & \sim 1/6 \\ 0 & \sim 2/3 \\ -1 & \sim 1/6 \end{cases}$$

虽然严格而言上述构造仍然不稀疏，但在计算上可以节省时间。可参考 [1].

针对 $d > 2^k$ 的情况，即原维度远大于降维后的维度时，我们有一个技巧可以减低计算复杂度。

以下面的随机矩阵构造为例。

$$A_{ij} = \begin{cases} +1 & \sim 50\% \\ -1 & \sim 50\% \end{cases}$$

此时计算 $B \cdot x = H_k \cdot C \cdot x$，其中 C 是一个稀疏矩阵，H_k 的每一列是一个可能的组合，即：

$$H_k = \begin{bmatrix} h_1 & h_2 & \cdots & h_{2^k} \end{bmatrix}, \quad H \in \mathbb{R}^{k \times 2^k}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad C \in \mathbb{R}^{2^k \times d}$$

C 的每一列是一个选择向量，上面给出一个例子，对 C 的要求是每列有且仅有一个 1，其余为 0。根据矩阵乘法的性质，我们可以得到计算开销变为 $\Theta(d)$，相比原有的 $\Theta(k \cdot d)$。

5 Fast JL 变换

回到我们的问题，我们希望有这样一个 JL 变换 $x \to Ax$，我们希望 A 是一个低维稀疏矩阵，但同时希望 $\|Ax\|^2$ 与 $\|x\|^2$ 大概率接近。这样就可以节约计算开销。但如果 A 就是一个低维稀疏矩阵，那么可以证明，当 x 的坐标分布非常不平衡时（少数坐标的取值占比太大），有很大概率无法保距。具体来说，有如下观察：

Claim 5.1. *对于给定的 $y \in \mathbb{R}^d$ with $\|y\|_2 = 1$,*

$$\|y\|_\infty \le \lambda = \sqrt{\frac{2\ln(4d/\delta)}{d}}.$$

令 A 为 $t \times d$ 采样矩阵，$t = 2\ln(4d/\delta)^2 \ln(4/\epsilon)/\epsilon^2$. 则 $\Pr\left[\|Ay\|_2^2 \notin (1-\epsilon, 1+\epsilon)\right] \le \delta/2$.

[2] 通过矩阵旋转来实现矩阵的稀疏化。

5.1 构造方法

$$\Phi = \mathrm{P} \cdot \mathrm{H} \cdot \mathrm{D}$$

P:

$$P \in \mathbb{R}^{k \times d}, \quad p_{ij} = \begin{cases} N(0, \frac{1}{q}) & \sim q \\ 0 & \sim 1-q \end{cases} , q = min\{\Theta(\frac{\log^2 n}{d}), 1\}$$

可以计算 P 的稀疏程度:

$$\#\{\text{non-zero of P}\} = k \cdot d \cdot q = k\log^2 n \ll kd$$

H:

$$H \in \mathbb{R}^{d \times d} : \text{H 是归一化的 Hadamard 矩阵}$$

Hadamard 矩阵的维度是 2 的幂次方，对于 $2^{k-1} < d < 2^k$，将剩余维度补 0 即可。Hadamard 矩阵满足 $H_d^T H_d = dI$，即 Hadamard 矩阵是正交矩阵。使用归一化的 Hadamard 矩阵后 $H^T H = I$

Hadamard 矩阵满足递推关系:

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}, \quad H = \frac{1}{\sqrt{d}} H_d$$

根据该递推式的性质，Hadamard 矩阵乘法可以进行加速:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad x_1, x_2 \in \mathbb{R}^{d/2}, \quad Hx = \frac{1}{\sqrt{d}} \begin{bmatrix} H_{d/2} \cdot x_1 + H_{d/2} \cdot x_2 \\ H_{d/2} \cdot x_1 - H_{d/2} \cdot x_2 \end{bmatrix}$$

$$T(d) = 2T(d/2) + \Theta(d) \Rightarrow T(d) = \Theta(d \log d)$$

D 是一个对角方阵，只有在对角线上的元素有非零值，且是 ± 1，满足:

$$D \in \mathbb{R}^{d \times d}, \quad D_{ii} = \begin{cases} +1 & \sim 0.5 \\ -1 & \sim 0.5 \end{cases}$$

5.2 启发性展示

Fast JL 变换的证明较为复杂，相关内容的证明可以参考 [2], 作者之一 B. Chazelle 的另一个天才工作是解决了欧氏空间最小生成树的线性算法 [3]. 我们在此给出一些启发性的说明，并进行计算复杂度分析。

Fast JL 变换构造的变换中，可以分为两部分：P 和 (H · D)。前者的构造和 n 有关，而后者与 n 无关，其计算可以通过预处理得到。

P 的作用和前文中的 A 一致，可以证明，如果 x 比较"均匀"，那么无须经过 HD 连乘。相应的，反例就是 x 向量的多数维度，比如 d-1 个维度都为 0，只有一个维度非零，那么此时 P 的多数变量未参与计算（有效的估计数变少），这是我们不希望的。而 HD 的作用就是让 x 以较大概率变得稠密（无论其原本是稀疏还是稠密），因此两者结合，会取得较好的效果。

计算复杂度如下，注意，最外的 n 同样是对于 n 个点应用变换导致的。$\Theta(d \log d)$ 来自于 (H · D) 的计算，$\Theta(|P|)$ 来自于 P · (∗) 的计算，|P| 指矩阵 P 的非零元素个数。$\widetilde{\Theta}$ 表示忽略次线性项中的指数项。

$$
\begin{aligned}
T(d, n) &= (\Theta(d \log d) + \Theta(|\mathrm{P}|))n \\
&= \Theta(d \log d + k \cdot d \cdot q) \cdot n \\
&= \Theta(d \log d + \frac{\log^3 n}{\epsilon^2}) \cdot n \\
&= \widetilde{\Theta}(d + \frac{1}{\epsilon^2})n
\end{aligned}
$$

相比于先前 JL 变换的复杂度 $\Theta(\frac{nd}{\epsilon^2} \log n) = \widetilde{\Theta}(\frac{d}{\epsilon^2})n$，有显著的加速效果。

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. Special Issue on PODS 2001.

[2] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Symposium on the Theory of Computing*, 2006.

[3] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM (JACM)*, 47(6):1028–1047, 2000.

[4] Wikipedia contributors. Rigid transformation — Wikipedia, the free encyclopedia, 2025. [Online; accessed 20-March-2025].