

5.4.2 Sketching-Algorithmus für F_2

Sketching-Algorithmus: Datenstromalgorithmus, der komplette Eingabe liest, zur Unterscheidung von Sampling-Algorithmen (nur informeller Begriff).

Arbeit: Alon, Mathias, Szegedy (1996).

Idee für F_2 -Algorithmus:

Randomisierte Projektionen. Bereits bei SimHash gesehen.
Allgemein:

- Datenvektor aus „hochdimensionalem“ Raum \mathbb{R}^d multiplizieren mit zufälliger $k \times d$ -Matrix.
- Liefert komprimierten Vektor in „niedrigdimensionalem Raum“ \mathbb{R}^k .
- Sicherstellen, dass Norm approximativ erhalten bleibt.

Später mehr dazu. Hier spezielle Variante für $k = 1$.

F_2 -Algorithmus idealisiert:

Eingabedatenstrom über Universum $U = \{1, \dots, m\}$.

Absolute Häufigkeiten f_1, \dots, f_m , $f := [f_1, \dots, f_m]^T$.

- Sei $Z = [Z_1, \dots, Z_m]^T \in \{-1, 1\}^m$ zufällig gleichverteilt gewählt. Dann berechne randomisierte Projektion

$$f \mapsto \langle f, Z \rangle = \sum_{i=1}^m f_i \cdot Z_i =: X.$$

- Ausgabe $Y := X^2$.

Das ist alles!

Tatsächlich ist $EY = F_2$. Später: Relativer Fehler höchstens 300 % mit Wahrscheinlichkeit mindestens $7/9 > 1/2$.

Probability-Amplification $\rightarrow (\varepsilon, \delta)$ -Approximation.

Bevor wir das beweisen:

Effiziente Realisierung des F_2 -Algorithmus:

- Wie kommen wir an die absoluten Häufigkeiten?

Bevor wir das beweisen:

Effiziente Realisierung des F_2 -Algorithmus:

- Wie kommen wir an die absoluten Häufigkeiten?

Antwort: Gar nicht. Addiere für jedes neue Element $j \in U$ zugehöriges Z_j zur Gesamtsumme. Dann am Ende für alle Vorkommen von j Gesamtbeitrag $f_j \cdot Z_j$.

Bevor wir das beweisen:

Effiziente Realisierung des F_2 -Algorithmus:

- Wie kommen wir an die absoluten Häufigkeiten?

Antwort: Gar nicht. Addiere für jedes neue Element $j \in U$ zugehöriges Z_j zur Gesamtsumme. Dann am Ende für alle Vorkommen von j Gesamtbeitrag $f_j \cdot Z_j$.

- Dann brauchen wir aber einen Zufallsvektor $Z \in \{-1, 1\}^m$ und wahlfreien Zugriff darauf.

Bevor wir das beweisen:

Effiziente Realisierung des F_2 -Algorithmus:

- Wie kommen wir an die absoluten Häufigkeiten?

Antwort: Gar nicht. Addiere für jedes neue Element $j \in U$ zugehöriges Z_j zur Gesamtsumme. Dann am Ende für alle Vorkommen von j Gesamtbeitrag $f_j \cdot Z_j$.

- Dann brauchen wir aber einen Zufallsvektor $Z \in \{-1, 1\}^m$ und wahlfreien Zugriff darauf.

Antwort: Problem haben wir schon in Abschnitt 5.2 gelöst. Hier reicht nämlich (Analyse) Z mit 4-fach unabhängigen Komponenten.

Effiziente Realisierung des F_2 -Algorithmus (Forts.):

Basierend auf der polynomiellen Körperklasse mit Polynomen vom Grad 3 liefert Satz 5.11 folgendes Verfahren für Generierung von Zufallsvektor $Z \in \{-1, 1\}^m$:

- Wähle $S \in \{0, 1\}^\ell$, $\ell = \lceil \log m \rceil$ zufällig gleichverteilt und irreduzibles Polynom über \mathbb{F}_2 vom Grad ℓ .

Kann beides mit $O(\log m)$ Bits abspeichern.

- Es gibt Funktionen $h_1, \dots, h_m: \{0, 1\}^\ell \rightarrow \{0, 1\}$, sodass $h_1(S), \dots, h_m(S)$ 4-fach unabhängig.

Funktionen können auf Platz $O(\log m)$ Platz und in Zeit $O(\log m \cdot \text{polylog } m)$ ausgewertet werden.

Algorithmus RANDPROJECT für F_2 :

Intialisierung:

Für $\ell := \lceil \log m \rceil$ wähle $S \in \{0, 1\}^\ell$ zufällig gleichverteilt und wähle irreduzibles Polynom über \mathbb{F}_2 vom Grad ℓ .
Setze $x := 0$.

Update für $a \in U$:

$$x := x + h_a(S).$$

Ausgabe:

$$y = x^2.$$

Satz 5.18:

Seien $\varepsilon, \delta > 0$. Dann liefern $r = \Theta((1/\varepsilon^2) \log(1/\delta))$ Kopien des Algorithmus RANDPROJECT eine (ε, δ) -Approximation von F_2 . Dies stellt 1-Runden-Datenstromalgorithmus dar mit Speicherplatz $O(r(\log m + \log n))$ und Zeit pro Update $O(r(\log m \text{ polyloglog } m + \log n))$.

Beweis:

Ressourcen: Klar mit Vorbetrachtungen.

Erwartungswert:

$$\begin{aligned} EY &= E(X^2) = E\left(\left(\sum_{i=1}^m f_i Z_i\right)^2\right) \\ &= \sum_{i=1}^m f_i^2 \cdot \underbrace{E(Z_i^2)}_{=1} + 2 \sum_{1 \leq i < j \leq m} f_i f_j \cdot \underbrace{E(Z_i Z_j)}_{=(EZ_i)(EZ_j) = 0} \\ &= F_2. \end{aligned}$$

Varianz: $V(Y) = E(Y^2) - (EY)^2$, also $E(Y^2)$ ausrechnen.

Benutze 4-fache Unabhängigkeit der Z_1, \dots, Z_m :

$E(Z_{i_1}Z_{i_2}Z_{i_3}Z_{i_4}) = 0$, falls mindestens ein Wert unter den Indizes i_1, i_2, i_3, i_4 genau einmal vorkommt.

Damit:

$$\begin{aligned} E(Y^2) &= E\left(\left(\sum_{i=1}^m f_i Z_i\right)^4\right) \\ &= \sum_{i=1}^m f_i^4 \cdot E(Z_i^4) + \binom{4}{2} \sum_{1 \leq i < j \leq m} f_i^2 f_j^2 \cdot E(Z_i^2 Z_j^2) \\ &= \sum_{i=1}^m f_i^4 + 6 \sum_{1 \leq i < j \leq m} f_i^2 f_j^2. \end{aligned}$$

Hatten: $E(Y^2) = \sum_{i=1}^m f_i^4 + 6 \sum_{1 \leq i < j \leq m} f_i^2 f_j^2.$

Außerdem:

$$(EY)^2 = F_2^2 = \sum_{i=1}^m f_i^4 + 2 \sum_{1 \leq i < j \leq m} f_i^2 f_j^2.$$

Damit:

$$V(Y) = E(Y^2) - (EY)^2 = 4 \sum_{1 \leq i < j \leq m} f_i^2 f_j^2 \leq 2F_2^2.$$

Tschebyscheff liefert:

$$\Pr\{|Y - F_2| \geq \varepsilon \cdot F_2\} \leq \frac{2}{\varepsilon^2},$$

z. B. $\varepsilon = 3 \rightarrow$ Fehlschlagswahrscheinlichkeit höchstens $2/9$.

Probability-Amplification mit $r = \Theta((1/\varepsilon^2) \log(1/\delta))$ Kopien des Schätzers Y liefert (ε, δ) -Approximation von F_2 . \square

5.4.3 Sketching-Algorithmen für F_k

Hier $k \in \mathbb{R}$, $k \geq 1$ ($k = 1$ der Vollständigkeit halber).

Erste Idee für Algorithmus:

Will naive Lösung mit m Zählern verbessern:

Ermittle für $X \in \{1, \dots, m\}$ zufällig gleichverteilt

mit nur *einem* Zähler exakt absolute Häufigkeit f_X .

5.4.3 Sketching-Algorithmen für F_k

Hier $k \in \mathbb{R}$, $k \geq 1$ ($k = 1$ der Vollständigkeit halber).

Erste Idee für Algorithmus:

Will naive Lösung mit m Zählern verbessern:

Ermittle für $X \in \{1, \dots, m\}$ zufällig gleichverteilt mit nur *einem* Zähler exakt absolute Häufigkeit f_X .

Dann:

$$E(m \cdot f_X^k) = m \cdot \sum_{i=1}^m \underbrace{E(f_X^k | X = i)}_{= f_i^k} \cdot \underbrace{\Pr\{X = i\}}_{= 1/m} = \sum_{i=1}^m f_i^k = F_k.$$

5.4.3 Sketching-Algorithmen für F_k

Hier $k \in \mathbb{R}$, $k \geq 1$ ($k = 1$ der Vollständigkeit halber).

Erste Idee für Algorithmus:

Will naive Lösung mit m Zählern verbessern:

Ermittle für $X \in \{1, \dots, m\}$ zufällig gleichverteilt mit nur *einem* Zähler exakt absolute Häufigkeit f_X .

Dann:

$$E(m \cdot f_X^k) = m \cdot \sum_{i=1}^m \underbrace{E(f_X^k | X = i)}_{= f_i^k} \cdot \underbrace{\Pr\{X = i\}}_{= 1/m} = \sum_{i=1}^m f_i^k = F_k.$$

Fein, ein erwartungstreuer Schätzer!

5.4.3 Sketching-Algorithmen für F_k

Hier $k \in \mathbb{R}$, $k \geq 1$ ($k = 1$ der Vollständigkeit halber).

Erste Idee für Algorithmus:

Will naive Lösung mit m Zählern verbessern:

Ermittle für $X \in \{1, \dots, m\}$ zufällig gleichverteilt mit nur *einem* Zähler exakt absolute Häufigkeit f_X .

Dann:

$$E(m \cdot f_X^k) = m \cdot \sum_{i=1}^m \underbrace{E(f_X^k | X = i)}_{= f_i^k} \cdot \underbrace{\Pr\{X = i\}}_{= 1/m} = \sum_{i=1}^m f_i^k = F_k.$$

Fein, ein erwartungstreuer Schätzer!

Leider: Z. B. für $(f_1, \dots, f_m) = (n, 0, \dots, 0)$:

Mit Wskt. $1 - 1/m$ relativer Fehler 100%. :- (

Zweiter Versuch:

Gleichverteilte Wahl des gezählten Elementes trotz evtl. ungleichmäßig verteilten Häufigkeiten schlecht. Will: Proportional zu Häufigkeit wählen.

Zweiter Versuch:

Gleichverteilte Wahl des gezählten Elementes trotz evtl. ungleichmäßig verteilten Häufigkeiten schlecht. Will: Proportional zu Häufigkeit wählen.

Das geht sogar:

Für Datenstrom $a = (a_1, \dots, a_n)$ wähle Index $X \in \{1, \dots, n\}$ zufällig gleichverteilt und zähle Vorkommen des Wertes a_X im Gesamtstrom.

Zweiter Versuch:

Gleichverteilte Wahl des gezählten Elementes trotz evtl. ungleichmäßig verteilten Häufigkeiten schlecht. Will: Proportional zu Häufigkeit wählen.

Das geht sogar:

Für Datenstrom $a = (a_1, \dots, a_n)$ wähle Index $X \in \{1, \dots, n\}$ zufällig gleichverteilt und zähle Vorkommen des Wertes a_X im Gesamtstrom.

Leider nicht mehr klar, wie das als 1-Runden-Datenstromalgorithmus realisierbar.

Zweiter Versuch:

Gleichverteilte Wahl des gezählten Elementes trotz evtl. ungleichmäßig verteilten Häufigkeiten schlecht. Will: Proportional zu Häufigkeit wählen.

Das geht sogar:

Für Datenstrom $a = (a_1, \dots, a_n)$ wähle Index $X \in \{1, \dots, n\}$ zufällig gleichverteilt und zähle Vorkommen des Wertes a_X im Gesamtstrom.

Leider nicht mehr klar, wie das als 1-Runden-Datenstromalgorithmus realisierbar.

Idee von Alon, Mathias und Szegedy: Wähle X wie oben, aber zähle Vorkommen von a_X nur im *Restdatenstrom* ab Position X .

Dritter und letzter Versuch:

Für Eingabe-Datenstrom $a = (a_1, \dots, a_n)$:

- Wähle $X \in \{1, \dots, n\}$ zufällig gleichverteilt.
- Zähle Vorkommen von Wert a_X an Positionen X, \dots, n .

Sei $r_i := |\{j \mid a_j = a_i, i \leq j \leq n\}|$. Es ist

$$\begin{aligned} E(r_X^k) &= \sum_{i=1}^m \sum_{j=1}^{f_i} E(r_X^k \mid X = \text{„}j\text{-te Position von Wert } i\text{“}) \cdot \frac{1}{n} \\ &= \frac{1}{n} (f_1^k + (f_1 - 1)^k + \dots + 2^k + 1^k + \\ &\quad f_2^k + (f_2 - 1)^k + \dots + 2^k + 1^k + \\ &\quad \dots + \\ &\quad f_m^k + (f_m - 1)^k + \dots + 2^k + 1^k). \end{aligned}$$

Extrahiere Teilsumme F_k durch Teleskopsummierung...

Benutze dazu Schätzer

$$Y := n(r_X^k - (r_X - 1)^k).$$

Dann:

$$\begin{aligned} EY &= \sum_{i=1}^m \sum_{j=1}^{f_i} E(Y \mid X = \text{„}j\text{-te Position von Wert } i\text{“}) \cdot \frac{1}{n} \\ &= \frac{n}{n} \sum_{i=1}^m \sum_{j=1}^{f_i} (j^k - (j-1)^k) \\ &= (f_1^k - (f_1 - 1)^k) + \dots + (2^k - 1^k) + 1^k + \dots + \\ &\quad (f_m^k - (f_m - 1)^k) + \dots + (2^k - 1^k) + 1^k \\ &= f_1^k + \dots + f_m^k = F_k. \end{aligned}$$

Auch Varianz ist gut, später.

Algorithmus SAMPLECOUNT für F_k :

Für Eingabe-Datenstrom $a = (a_1, \dots, a_n)$:

- Wähle $X \in \{1, \dots, n\}$ zufällig gleichverteilt.
- Zähle Vorkommen r_X von Wert a_X an Positionen X, \dots, n
- Ausgabe $Y = n(r_X^k - (r_X - 1)^k)$.

Stichprobe realisieren mit Reservoir-Sampling.

Satz 5.19:

Sei $k \in \mathbb{R}$ mit $k \geq 1$, $\varepsilon, \delta > 0$ und $\ell := \min\{m, n\}$. Dann liefern $r = O((1/\varepsilon^2) \log(1/\delta) k \ell^{1-1/k})$ Kopien von SAMPLECOUNT eine (ε, δ) -Approximation von F_k . Dies ist ein 1-Runden-Datenstromalgorithmus mit Speicherplatzbedarf und Zeit für Updates $O(r \cdot (\log m + \log n))$.

Beweis:

Ressourcen:

r Kopien von Reservoir-Sampling mit Stichprobengröße 1,
pro Kopie $O(\log m + \log n)$ Platz / Zeit pro Element.

Erwartungswert: Bereits erledigt.

Varianz: Analog zu Formel für EY :

$$\begin{aligned} E(Y^2) &= \sum_{i=1}^m \sum_{j=1}^{f_i} E(Y^2 \mid X = \text{„}j\text{-te Position von Wert } i\text{“}) \cdot \frac{1}{n} \\ &= \frac{n^2}{n} \sum_{i=1}^m \sum_{j=1}^{f_i} (j^k - (j-1)^k)^2. \end{aligned}$$

Fakt: Für beliebige $x, y \in \mathbb{R}$ mit $x \leq y$ und $k \in \mathbb{R}$ mit $k \geq 1$:
$$y^k - x^k \leq (y - x)ky^{k-1}.$$

Hatten:
$$E(Y^2) = \frac{n^2}{n} \sum_{i=1}^m \sum_{j=1}^{f_i} (j^k - (j-1)^k)^2.$$

Anwenden auf des Fakts auf jeweils einen der beiden Faktoren von $(j^k - (j-1)^k)^2$:

$$\begin{aligned} E(Y^2) &\leq n \sum_{i=1}^m \sum_{j=1}^{f_i} kj^{k-1} (j^k - (j-1)^k) \\ &\leq n \sum_{i=1}^m \sum_{j=1}^{f_i} kf_i^{k-1} (j^k - (j-1)^k) \\ &= kn \sum_{i=1}^m f_i^{2k-1} = kn \cdot F_{2k-1}. \end{aligned}$$

Noch ein Fakt: Für $x \in \mathbb{R}^d$, $r \geq s \geq 1$: $\|x\|_r \leq \|x\|_s$.

Hatten:

$$E(Y^2) = kn \cdot F_{2k-1}.$$

Sei $\ell := \min\{m, n\}$. Dann:

$$\begin{aligned} F_{2k-1} &\stackrel{\text{Fakt}}{\leq} F_k^{(2k-1)/k} = F_k^{2-1/k} \\ &\leq F_k^2 \cdot \left(\frac{n^k}{\ell^{k-1}}\right)^{-1/k} = F_k^2 \cdot \frac{\ell^{1-1/k}}{n}. \end{aligned}$$

Insgesamt:

$$V(Y) \leq E(Y^2) \leq kn \cdot F_{2k-1} \leq k\ell^{1-1/k} \cdot F_k^2.$$

Tschebyscheff:

$$\Pr\{|Y - F_k| \geq \varepsilon F_k\} \leq \frac{k\ell^{1-1/k}}{\varepsilon^2}.$$

Behauptung mit Probability-Amplification. □

Problem: Dimensionsreduktion

- Gegeben Punkte $v_1, \dots, v_n \in \mathbb{R}^d$ („hochdimensionaler Raum“).
- Ziel: Abbilden auf $\tilde{v}_1, \dots, \tilde{v}_n \in \mathbb{R}^k$, $k \ll d$ („niedrigdimensionaler Raum“), sodass Abstände nicht zu sehr verzerrt.

Passende Abbildung: *Metrische Einbettung*.

Abstände: Für Vektor $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$:

- *L_p -Normen, $p > 0$:* $\|\mathbf{x}\|_p := \left(\sum_{i=1}^d |x_i|^p \right)^{1/p}$.
- *Maximumsnorm, L_∞ :* $\|\mathbf{x}\|_\infty := \max_{1 \leq i \leq d} |x_i|$.
- *Hammingnorm, L_0 :* $\|\mathbf{x}\|_0 := |\{i \mid x_i \neq 0\}|$.

Anwendungen:

Dimensionsreduktion:

- Clustering;
- algorithmisches Lernen;
- Approximate Nearest-Neighbor Search (ANN);
- Datenstromalgorithmen, z. B.
für Normen und Abstände (hier).

Metrische Einbettungen allgemein:

Approximationsalgorithmen für Graphprobleme.

5.5.1 Stabile Verteilungen

Fakt: Linearkombinationen von unabhängigen Kopien normalverteilter ZV wieder normalverteilt.

Definition 5.20:

Seien X_1, \dots, X_d unabhängige Kopien von reeller ZV X mit folgender Eigenschaft. Für ein $p \in \mathbb{R}^+$ gelte für beliebige $a = (a_1, \dots, a_d) \in \mathbb{R}^d$, dass

$$a_1 X_1 + \dots + a_d X_d \sim \|a\|_p \cdot X,$$

wobei $\|a\|_p = \left(\sum_{i=1}^d |a_i|^p\right)^{1/p}$ und $Z \sim Z'$ bedeutet,

dass Z und Z' identisch verteilt. Nenne dann Verteilung von X *p-stabil*.

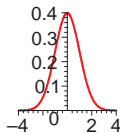
Satz 5.21:

Es gibt p -stabile Verteilungen für beliebige $p \in (0, 2]$.

Beispiele für stabile Verteilungen:

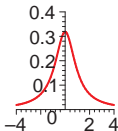
- *Normalverteilung*: 2-stabil.

$$f_N(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad x \in \mathbb{R}.$$



- *Cauchyverteilung*: 1-stabil.

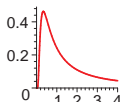
$$f_C(x) = \frac{1}{\pi} \cdot \frac{\gamma}{\gamma^2 + (x - \delta)^2}, \quad x \in \mathbb{R}.$$



- *Lévyverteilung*: 1/2-stabil.

$$f_L(x) = \frac{\gamma}{2\pi} \cdot \frac{1}{(x - \delta)^{3/2}} \exp\left(\frac{-\gamma}{2(x - \delta)}\right),$$

$x > \delta.$



Diese und $x \mapsto f_L(-x)$ einzige Familien von stabilen Verteilungen mit bekannter geschlossener Form.

Eigenschaften stabiler Verteilungen:

- Für $p \in (0, 2)$ gilt: p -stabile Verteilung ist approximativ Potenzgesetzverteilung mit Exponent $1 + p$:
Dichtefunktion sei f , dann existiert $c > 0$ geeignet, sodass $f(x)/(cx^{-(1+p)}) \rightarrow 1$ für $x \rightarrow \infty$.
Damit Skaleninvarianz und „heavy tails“.
- Für $1 < p < 2$ existiert Erwartungswert, aber Varianz nicht.
- Für $p \leq 1$ existiert weder Erwartungswert, noch Varianz.

(Ohne Beweise.)

Simulation von stabilen Verteilungen:

Für Normalverteilung: Box-Muller-Verfahren.

Allgemeines Verfahren:

Sei F invertierbare Verteilungsfunktion.

- Wähle $U \in [0, 1]$ zufällig gleichverteilt.
- Ausgabe $F^{-1}(U)$.

Korrektheit klar:

$$\Pr\{F^{-1}(U) \leq x\} = \Pr\{U \leq F(x)\} = F(x).$$

Simulation von stabilen Verteilungen (Forts.):

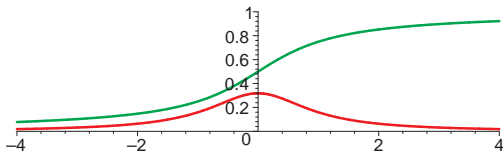
Beispiel: Normalisierte Cauchyverteilung

$$\text{Dichte } f(x) = \frac{1}{\pi} \cdot \frac{1}{1+x^2},$$

$$\text{Verteilungsfunktion } F(x) = \arctan(x)/\pi + 1/2.$$

$$\text{Damit: } F^{-1}(y) = \tan(\pi(y - 1/2)), y \in (0, 1).$$

Für Implementierung Approximation mit endlicher Genauigkeit (Standard-Numerik-Tricks, Taylorreihen...).



Simulation von stabilen Verteilungen (Forts.):

Allgemein gilt:

Satz 5.22:

Seien U_1 und U_2 über $[0, 1]$ gleichverteilte, unabhängige Zufallsvariablen. Dann ist für $p \in (0, 2]$ die im Folgenden definierte Zufallsvariable $S(U_1, U_2, p)$ p -stabil.

Sei $\theta := \pi(U_1 - 1/2)$ und

$$S(U_1, U_2, p) := \begin{cases} \frac{\sin(p\theta)}{(\cos \theta)^{1/p}} \left(\frac{\cos(\theta(1-p))}{-\ln U_2} \right)^{(1-p)/p}, & p \neq 1; \\ \tan \theta, & p = 1. \end{cases}$$

Für Cauchyverteilung bewiesen, ansonsten hier nicht.

Für $p = 2$ Formel wie beim Box-Muller-Verfahren.

5.5.2 Randomisierte Projektionen

Wichtiges positives Ergebnis für L_2 -Norm:

Satz 5.23 (Johnson-Lindenstrauss-Lemma, 1984):

Gegeben Punkte v_1, \dots, v_n in \mathbb{R}^d und $\varepsilon > 0$. Dann existiert approximative metrische Einbettung in \mathbb{R}^k mit relativem Fehler ε , d. h., $\tilde{v}_1, \dots, \tilde{v}_n \in \mathbb{R}^k$ mit

$$(1 - \varepsilon) \|v_i - v_j\|_2 \leq \|\tilde{v}_i - \tilde{v}_j\|_2 \leq (1 + \varepsilon) \|v_i - v_j\|_2$$

und $k = O((1/\varepsilon^2) \log n)$.

Und: Kann das auch noch effizient berechnen. . .

Realisierung durch randomisierte Projektionen:

- Benutze $k \times d$ -Matrix X mit zufälligen Einträgen. Einträge unabhängig voneinander.
- Einbettung:

$$v \mapsto \tilde{v} := \frac{1}{\sqrt{k}}Xv.$$

(Projektion auf den k -dimensionalen Unterraum in \mathbb{R}^n , der von den Zeilen von X aufgespannt wird.)

Wie Einträge von X wählen?

Wahl der Matrix X für randomisierte Projektion:

- Klassisch: Einträge gemäß $N(0, 1)$ -Verteilung.
(Vgl. SimHash-Verfahren.)
- Diskret & effizient: Einträge gleichverteilt aus $\{-1, 1\}$.
(Vgl. praktisches SimHash-Verfahren, F_2 -Algorithmus.)

Kann zeigen:

Satz 5.24:

Sei $\beta, \varepsilon > 0$. Sei X eine randomisierte Matrix wie oben beschrieben. Dann ist Fehlerschranke in JL-Lemma mit Wahrscheinlichkeit $1 - 1/n^\beta$ erfüllt für $k = O((\beta/\varepsilon^2) \log n)$.

Strategie für Beweis (Fall $k = 1$):

Gemäß Def. 5.20 für Zufallsvektor X , Komponenten unabhängig gemäß 2-stabiler Verteilung $N(0, 1)$:

$$E(\langle X, v \rangle^2) = E(\|v\|_2^2 \cdot N(0, 1)^2) = \|v\|_2^2.$$

Für Gleichverteilung auf $\{-1, 1\}$ dasselbe,
Beweis siehe Analyse F_2 -Algorithmus RANDPROJECT.

Zeige nun weiter, dass sogar starke Konzentration um Erwartungswert gilt (exponentielle Verbesserung des Fehlers in spendierter Anzahl Dimensionen).

Bemerkung:

In schwächeren Form bei F_2 -Algorithmus gesehen. Für Ergebnis hier bessere Analyse wie bei Beweis der Chernoff-Schranken erforderlich.

Randomisierte Projektionen für beliebige p -Normen, $p \in (0, 2)$:

Nur für Spezialfall $k = 1$.

Wieder zufälliger Vektor $X \in \mathbb{R}^d$,
Komponenten unabhängig gemäß p -stabiler Verteilung D .

Gemäß Definition 5.20:

$$\langle X, v \rangle \sim \|v\|_p \cdot D.$$

Problem für Anwendung bei Dimensionsreduktion:

- Eventuell ist Erwartungswert oder Varianz unendlich.
Abhilfe: Erwartungswert \rightarrow Median.
- Kann für $p = 1$, L_1 -Norm, zeigen:
Jede approximative Einbettung mit konstantem Fehler benötigt $k = n^{\Omega(1)}$ Dimensionen.