Unapređenje SVD algoritma

STUDENT
DANIJEL RUJEVIĆ
1095/2013

PROFESOR
DR MLADEN NIKOLIĆ
ASISTENT
DR STEFAN MIŠKOVIĆ

OPIS

Jedna od najkorisnijih dekompozicija je dekompozicija singularnih vrednosti (SVD - *singular value decomposition*). Ona nam pruža mogućnost rešavanja ili razumevanja problema koji su definisani sistemima jednačina zasnovanih na sigularnim ili blisko singularnim matricama. Dok druge metode ne pokazuju dobre rezultate u tim slučajevima.

Ovaj metod ima široku primenu:

1. rešavanje linearnih jednačina,
2. za kompresiju podataka,
3. u algoritmu za preporučivanje,
4. za konstrukciju frekvencije reči u dokumentima,
5. za računanje $2 - \boldsymbol{norme}$ itd.

Ne zahteva da matrica A koju faktorišemo bude kvadratna. Za A dimenzija $p \times q$ postoji ortogonalna matrica U dimenzija $p \times p$, dijagonalna matrica Σ dimenzija $p \times q$ i ortogonalna matrica V dimenzija $q \times q$

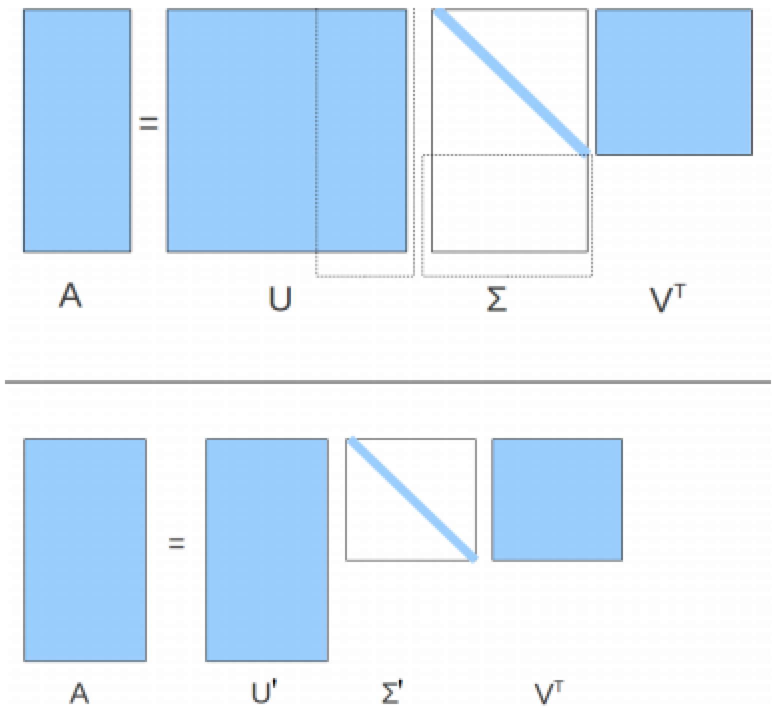
$$A = U\Sigma V^T$$

Predstavlja formulu SVD, međutim moguće je predstaviti je i sledećom formulom:

$$X = U'\Sigma' V^T$$

gde su U dimenzija $p \times q$, Σ dimenzija $q \times q$ i V dimenzija $q \times q$. Ovakva formula predstavlja tanku SVD.

Sledeća slika ilistruje formule.



Σ se u računarstvu najčešće posmatra kao vektor dužine q, a formula može biti predstavljena kao suma matrica ranka-1 $\sum_i u_i \sigma_i v_i^T$, gde σ_i predstavljaju singularne vrednosti (sa diagonale Σ), u_i i v_i su singularni vektori odnosno kolone U i V

$$A = \sum_{i=1} \sigma_i u_i v_i^T = \qquad + \qquad + \cdots .$$

Važi da su:

1. Kolone matrice U su sopstveni vektori AA^T - levi singularni vektori matrice A
2. Kolone matrice V su sopstveni vektori A^TA - desni singularni vektori matrice A
3. Dijagonalni elementi Σ - singularne vrednosti matrice A
4. Rang matrice A je broj $\sigma_i <> 0$

RAČUNANJE DEKOMPOZICIJE

Na osnovu prethodnog lako se može doći do sledećih formula:

$A^T A = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T$

$AV = U\Sigma$

$det(A^T A - \lambda I) = 0$ odavde je moguće izračunati V i Σ, a zatim uvrštavanjem u prethodnu formulu i U

UNAPREĐENJE ALGORITMA

Složenost izračunavanja

Izračunavanje potpune singularne dekompozicije je računarski izuzetno zahtevno i potrebno je $\Theta(pq^2)$ operacija. Cilj je komplseksnost smanjiti na $O(pqr)$. Jako brzo nakon pojavljivanja računara i praktičnog SVD algoritma počelo tragati za efikasnijim metodima. Tako je nastala tanka SVD modifikacija i slične metode koje su se odnosile na promenu podataka u matrici.

Predlaganje unapređenja

Ovim naučnim radom se predlažu sledeće modifikacije i unapređenja:

1. modifikacije sabiranja,
2. rank-1 modifikacije,
3. smanjenje kompleksnosti

UNAPREĐENJE ALGORITMA - NASTAVAK

Moguća redukcija

Ukoliko pretpostavimo da je redosled vrednosti σ_i u neopadajućem poretku možemo eliminisati male vrednosti tako da ogranicimo $i < r < q$. Tada je U dimenzija $p \times r$, σ vektor dimenzije r i V dimenzija $r \times r$ i ovim smo dobili tanku SVD ranka-r.

MODIFIKACIJA SABIRANJA

Ako je $X = USV^T$ onda za sabiranje želimo da postavimo modifikaciju tako da: $X + AB^T = \begin{bmatrix} U & A \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V & B \end{bmatrix}^T$. U slučaju kad je $rank(X + AB^T) \leqslant r + c < min(p, q)$, matrice U, V, A, B su visoke tanke.

Dalje ako je P ortogonalna baza $(I - UU^T)A$ i $R_A = P^T(I - UU^T)A$ onda

$$\begin{bmatrix} U & A \end{bmatrix} = \begin{bmatrix} U & P \end{bmatrix} \begin{bmatrix} I & U^T A \\ 0 & R_A \end{bmatrix}$$

Slično je za $QR_B = (I - VV^T)B$.

Kombinacijom ovih formula dobijamo $X + AB^T = \begin{bmatrix} U & P \end{bmatrix} K \begin{bmatrix} V & Q \end{bmatrix}^T$

$$\mathbf{K} \doteq \begin{bmatrix} \mathbf{I} & \mathbf{U}^\top \mathbf{A} \\ \mathbf{0} & \mathbf{R_A} \end{bmatrix} \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{V}^\top \mathbf{B} \\ \mathbf{0} & \mathbf{R_B} \end{bmatrix}^\top = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{U}^\top \mathbf{A} \\ \mathbf{R_A} \end{bmatrix} \begin{bmatrix} \mathbf{V}^\top \mathbf{B} \\ \mathbf{R_B} \end{bmatrix}^\top$$

Kada postavimo $U'^T K V' = S'$ tada K proširuje prodprostore a U' i V' rotiraju $\begin{bmatrix} U & P \end{bmatrix}$ i $\begin{bmatrix} V & Q \end{bmatrix}$

$$\mathbf{X} + \mathbf{A}\mathbf{B}^\top = ([\mathbf{U} \quad \mathbf{P}]\mathbf{U}')\mathbf{S}'([\mathbf{V} \quad \mathbf{Q}]\mathbf{V}')^\top$$

RANK-1 MODIFIKACJIE

Operation	Known	Desired	$\mathbf{a}$	$\mathbf{b}^\top$
Update	$\mathbf{U}\mathbf{S}[\mathbf{V}^\top \quad \mathbf{0}] = [\mathbf{X} \quad \mathbf{0}]$	$\mathbf{U}'\mathbf{S}'\mathbf{V}'^\top = [\mathbf{X} \quad \mathbf{c}]$	$\mathbf{c}$	$[0, \ldots, 0, 1]$
Downdate	$\mathbf{U}\mathbf{S}\mathbf{V}^\top = [\mathbf{X} \quad \mathbf{c}]$	$\mathbf{U}'\mathbf{S}'\mathbf{V}'^\top = \mathbf{X}$	$-\mathbf{c}$	$[0, \ldots, 0, 1]$
Revise	$\mathbf{U}\mathbf{S}\mathbf{V}^\top = [\mathbf{X} \quad \mathbf{c}]$	$\mathbf{U}'\mathbf{S}'\mathbf{V}'^\top = [\mathbf{X} \quad \mathbf{d}]$	$\mathbf{d} - \mathbf{c}$	$[0, \ldots, 0, 1]$
Recenter	$\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathbf{X}$	$\mathbf{U}'\mathbf{S}'\mathbf{V}'^\top = \mathbf{X}(\mathbf{I} - \frac{1}{q}\mathbf{1}\mathbf{1}^\top)$	$-\frac{1}{q}\mathbf{X}\mathbf{1}$	$\mathbf{1}^\top \doteq [1, \ldots, 1]$

RANK-1 MODIFIKACJIE - NASTAVAK

Ako uzmemo da važi

$$\mathbf{m} \doteq \mathbf{U}^\top \mathbf{a}; \quad \mathbf{p} \doteq \mathbf{a} - \mathbf{U}\mathbf{m}; \quad R_a = \|\mathbf{p}\|; \quad \mathbf{P} = R_a^{-1} \cdot \mathbf{p}$$

$$\mathbf{n} \doteq \mathbf{V}^\top \mathbf{b}; \quad \mathbf{q} \doteq \mathbf{b} - \mathbf{V}\mathbf{n}; \quad R_b = \|\mathbf{q}\|; \quad \mathbf{Q} = R_b^{-1} \cdot \mathbf{q}.$$

računanje K se svodi u slučaju proširivanja na

$$\mathbf{K} = \begin{bmatrix} \mathbf{S} & \mathbf{m} \\ \mathbf{0} & \|\mathbf{p}\| \end{bmatrix}$$

što može biti izvršeno u $O(r^2)$, a u slučaju smanjivanja na

$$\mathbf{K} = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \left(\mathbf{I} - \begin{bmatrix} \mathbf{S}\mathbf{n} \\ 0 \end{bmatrix} \begin{bmatrix} \mathbf{n} \\ \sqrt{1 - \mathbf{n}^\top \mathbf{n}} \end{bmatrix}^\top \right)$$

se može rešiti bez P i $Q = (b - Vn)/\sqrt{1 - n^T n}$ korišćenjem samo i-tog reda V

SMANJIVANJE KOMPLEKSNOSTI

Korišćenjem odogvarajuće implementacije prethodnih koraka, moguće je kompleksnost dodatno smanjiti. Tako je za operaciju dodavanja moguće postići umesto $O(p(r+c)^2)$ - $O(pr)$, za rediagojalizaciju umesto $O((r+c)^3)$ na $O(r^2)$, a za rotiranje bodprostora sa $O((p+q)(c+r)^2)$ na $O(r^3)$. Takva unapređenja se postižu umesto prethodne formule koristi pojednostavljena

$$\mathbf{U}_{p\times r} \cdot \mathbf{U}'_{r\times r} \cdot \mathbf{S}_{r\times r} \cdot \mathbf{V}'^{\top}_{r\times r} \cdot \mathbf{V}^{\top}_{q\times r}$$

i zatim se u svakom koraku umesto da se rotiraju U i V rotiraju značajno manje U' i V'

ALGORITAM

```python
def svd_upd(V, c):
    #prosirujemo V
    #V = np.vstack([V, np.zeros(V.shape[1])])
    #kreiramo b i punimo 0
    b = np.zeros(V.shape[0])
    #dodajemo 1 na kraj
    b[-1] = 1
    #transponujemo
    b = np.reshape(b, (b.shape[0], 1))
    #punimo a
    a = np.reshape(c, (-1, 1))
    return a,b
```

```python
def svd_rev(V,X, c):
    #prosirujemo V
    V = np.vstack([V, np.zeros(V.shape[1])])
    #kreiramo b i punimo 0
    b = np.zeros(V.shape[0])
    b[-1] = 1
    #transponujemo
    b = np.reshape(b, (b.shape[0], 1))
    #punimo a
    a = np.reshape(X[:,-1] - c, (-1, 1))
    return a,b
```

```python
def svd_down(V, X):
    #kreiramo b i punimo 0
    b = np.zeros(V.shape[0])
    b[-1] = 1
    #transponujemo
    b = np.reshape(b, (b.shape[0], 1))
    #punimo a
    a = np.reshape(np.multiply(X[:,-1], -1), (-1, 1))
    return a,b
```

```python
def svd_recenter(V, X):
    #kreiramo b i punimo 1
    ones = np.ones(V.shape[1])
    b = np.reshape(ones, (-1, 1))
    #parametri potrebni za a
    n = np.reshape(np.dot(np.transpose(V), b), (-1, 1))
    q = b - np.dot(V, n)
    #punimo a
    a = np.reshape(np.multiply((-1/q), np.dot(X, b)), (-1, 1))
    return a,b
```

ALGORITAM - NASTAVAK

```python
def rediagonalization(U,S,V,a,b):
    m = np.reshape(np.dot(np.transpose(U), a), (-1, 1))
    p = np.reshape(a - np.dot(U, m), (-1, 1))
    Ra = np.linalg.norm(p)
    P = np.reshape(np.multiply((1 / Ra), p), (-1, 1))
    n = np.reshape(np.dot(np.transpose(V), b), (-1, 1))
    q = b - np.dot(V, n)
    Rb = np.linalg.norm(q)
    Q = np.reshape(np.multiply((1 / Rb), q), (-1, 1))

    k = S
    K = np.zeros((k.shape[0] + 1, k.shape[0] + 1))
    K[:-1,:-1] = k
    stack = np.vstack(np.append(m, Ra))
    t = np.reshape(np.append(n, Rb), (1, -1))
    dot = np.dot(stack, t)
    K = np.add(K, dot)
    return K
```

ALGORITAM - NASTAVAK

```
#op predstavlja operaciju koja se izvrsava
#0-upd,1-dwn,2-rev,3-rec
def svd(U,S,V,X,c=None,op=0):
    if op==0:
        if type(c)==type(np.array([])):
            a, b = svd_upd(V, c)
        else:
            return None,None,None
    elif op==1:
        a, b = svd_down(V, X)
    elif op==2:
        if type(c)==type(np.array([])):
            a, b = svd_rev(V, X, c)
        else:
            return None,None,None
    elif op==3:
        a, b = svd_recenter(V, X)
    else:
        return None,None,None

    k=rediagonalization(U,S,V,a,b)

    Sn, Vn=np.linalg.eig(k)
    Sn=np.diag(Sn)
    Un=np.transpose(np.linalg.inv(Vn))

    return Un, Sn, Vn
```

KRAJ