# Appunti del corso di esercitazioni di Calcolo Scientifico a.a. 2009-2010 "SVD e compressione di immagini"

Dario A. Bini

20 novembre 2013

## 1 Introduzione

La decomposizione ai valori singolari è uno strumento che può essere usato per comprimere immagini digitali. Ricordiamo che nel formato PGM una immagine di $m \times n$ pixel in bianco e nero, o per essere più precisi, in scala di grigi viene memorizzata come una matrice $m \times n$ di numeri interi compresi tra 0 e 255 dove 0 corrisponde al nero, 255 al bianco e i valori intermedi corrispondono alle varie tonalità di grigio. Una immagine a colori viene memorizzata nel formato PNM con tre matrici di interi tra 0 e 255, la prima registra le intensità del rosso la seconda del verde e la terza del blu. La modalità è detta RGB dai tre colori Red Green Blue.

Si ricorda che una matrice $A$ di dimensioni $m \times n$ può essere rappresentata mediante la sua decomposizione ai valori singolari nota come SVD (Singular Values Decomposition) come

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

dove $U$ è una matrice $m \times m$ ortogonale, $V$è una matrice $n \times n$ ortogonale, $\Sigma$ è una matrice $m \times n$ diagonale, cioè tale che $(\Sigma)_{i,j} = 0$ se $i \neq j$, $(\sigma)_{i,i} = \sigma_i \geq 0$, $i = 1, \ldots, \min(m, n)$. Si assume inoltre che $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min m,n}$.

Le colonne di $U$ denotate con $\boldsymbol{u}_i$, $i = 1, \ldots, m$ sono i vettori singolari destri di $A$, le colonne $\boldsymbol{v}_i$ di $V$, per $i = 1, \ldots, n$ sono i vettori singolari sinistri di $A$ mentre i numeri $\sigma_i$ sono i valori singolari.

La decomposizione ai valori singolari può essere scritta in modo equivalente come

$$A = \sum_{i=1}^{\min(m,n)} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

ed è interessante la proprietà per cui

$$A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

è quella matrice che minimizza $||A - X||_2$ tra tutte le matrici $X$ di rango $k$. Cioè è la migliore approssimazione di rango $k$ di $A$ valutata in norma 2.

Se vogliamo quindi approssimare un'immagine $A$ di dimensione $m \times n$ pixel attraverso una matrice di rango $k << m, n$ possiamo considerare la matrice $A_k$ di migliore approssimazione in norma 2.

Possiamo allora cercare di ridurre lo spazio di memoria occupato da un'immagine $m \times n$ calcolando la SVD della matrice $A$ che la rappresenta e memorizzando solo i primi $k$ valori/vettori singolari. Vogliamo mostrare questa possibilità con una implementazione diretta.

Più precisamente svolgiamo la seguente elaborazione:

1. Calcoliamo la SVD $A = U\Sigma V^T$;
2. calcoliamo $\hat{\boldsymbol{u}}_i = \boldsymbol{u}_i * \sqrt{\sigma_i}$, $\hat{\boldsymbol{v}}_i = \boldsymbol{v}_i * \sqrt{\sigma_i}$
3. Scegliamo un valore di taglio $t << m, n$ e registriamo solo i primi $t$ vettori $\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{v}}_k$, $k = 1, \ldots, t$.
4. Per risparmiare ingombro di memoria non registriamo integralmente le componenti dei vettori $\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{v}}_k$ bensì registriamo la parte intera delle componenti moltiplicate per un fattore di scala $\gamma > 1$, ad esempio $\gamma = 20$.
5. Creeremo quindi un file contenente: i valori di $m, n, t$ seguiti dagli $(m+n)*t$ valori delle componenti $\hat{u}_{i,k}$ e $\hat{v}_{j,k}$ di $\hat{\boldsymbol{u}}_k$ e $\hat{\boldsymbol{v}}_k$.

Per ricostruire l'immagine rappresentata in questo modo è sufficiente:

1. leggere i valori di $m, n$ e $t$, e le componenti dei vettori $\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{v}}_k$, $k = 1, \ldots, t$
2. costruire $a_{i,j} = \gamma^{-2} \sum_{k=1}^{t} \hat{u}_{i,k} \hat{v}_{j,k}$
3. registrare tali valori in un file PGM.

Tenendo presente che l'ingombro di memoria di ogni singolo elemento è di circa un byte, l'ingombro del file compresso sarà di circa $(m + n)t$ byte contro $mn$ byte del file originale con un rapporto di compressione di un fattore $c = t(1/m + 1/n)$.

Si osserva che il valore di taglio lo possiamo scegliere direttamente in base al fattore di compressione $c$ che vogliamo ottenere, oppure in base alla qualità dell'immagine che vogliamo raggiungere. In questo secondo caso possiamo scegliere il taglio in modo che $\sigma_i/\sigma_1 < \epsilon$ per $i > t$ per un $\epsilon > 0$ fissato. Una scelta ragionevole di $\epsilon$, motivata dal fatto che i valori numerici dei pixel sono interi compresi tra 0 e 255, è ad esempio $\epsilon = 1/1000$.

Proveremo questa tecnica con l'immagine $2288 \times 1712$ da 4 mega pixel riportata in figura 2, che nel formato PGM in un file ASCII ha un ingombro di circa 14 mega byte. Scegliendo $t = 150$ ci aspettiamo un rapporto di compressione di un fattore 6.5.

Per implementare questa tecnica abbiamo bisogno di calcolare la SVD. Questo calcolo lo faremo una volta per tutte e memorizzeremo i risultati in un

file. Il calcolo della SVD lo facciamo col programma di LAPACK DGESV che utilizziamo mediante l'interfaccia dato dalla subroutine svd che mostriamo di seguito

```
! Calcola la SVD di una matrice A generica m*n.
! Se r=min(m,n), allora le prime r colonne di U conterranno
! i vettori singolari sinistri e le prime r righe di VT
! conterranno i vettori singolari destri.
! U e' una matrice m*m e VT e' una  matrice n*n.

SUBROUTINE CALCOLA_SVD(M,N,A,D,U,VT)
  implicit none
  integer, parameter :: dp=kind(0.d0)
  integer :: m, n, ldu, ldvt, info, lwork, lda
  real(dp)                       :: a(m,n), u(m,m), vt(n,n)
  real(dp), dimension(min(m,n)) :: d
  real(dp), dimension(3*(m+n))  :: work
  character :: jobu, jobvt

  jobu='S'
  jobvt='s'
  ldu=m
  ldvt=n
  lda=m
  lwork=3*(m+n)

  call DGESVD( JOBU,&! U contiene i vettori singolari sinistri
          JOBVT,&! V contiene i vettori singolari destri
          M, N,&! Dimensioni della matrice di lavoro
          A,&! Matrice di lavoro
          LDA,&
          D,&! Il vettore dei valori singolari
          U,&!la matrice dei v.s. sinistri
          LDU,&
          VT,&!la matrice dei v.s. destri
          LDVT, WORK, LWORK, INFO )
end subroutine
```

Il programma che segue legge la fotografia in formato pgm dal file `foto.pgm`, calcola la sua SVD e la salva nel file `full_svd.txt`. Nello scrivere il file si utilizza l'istruzione `write` assieme ad un formato di scrittura. Non si lascia cioè il formato libero poiché il compilatore inserirebbe troppi spazi che occupano memoria, ma scegliamo noi come scrivere i numeri. Infatti col codice di formato `E12.6` scriviamo con notazione esponenziale su 12 campi e lasciando 6 cifre dopo il punto decimale. Un numero con questo formato assume ad esempio la forma `-.123456E-23`.

```
program foto_svd
  implicit none
  integer, parameter                     :: dp=kind(0.d0)
  real(dp), dimension(:,:), allocatable :: a, u, vt
  real(dp), dimension(:), allocatable   :: d
  integer                                :: m, n, i, j, k, liv
  character :: ch
  open(unit=2, file='foto.pgm')
  read(2,*) ch
  read(2,*) n , m
  read(2,*) liv
  allocate(a(m,n))
  do i=1,m
     do j=1,n
        read(2,*) a(i,j)
     end do
  end do

  write(*,*)"calcolo la svd di una fotografia ",m,"x",n
  write(*,*)"L'operazione puo' richiedere alcuni minuti"
  allocate(u(m,m), vt(n,n), d(min(m,n)))
  call calcola_svd(m,n,a,d,u,vt)

  write(*,*)"Salvo la svd della foto nel file 'full_svd.txt'"
  open(unit=30, file="full_svd.txt")
  write(30,100)m,n
  do i=1,min(m,n)
     write(30,200)d(i)
  end do
  do i=1,m
     do k=1,min(m,n)
        write(30,200)u(i,k)
     end do
  end do
  do j=1,n
     do k=1,min(m,n)
        write(30,200)vt(k,j)
     end do
  end do
100 format(2I5)
200 format(E12.6)
end program
```

Avendo scritto la subroutine `calcola_svd` nel file `interfaccia_svd.f90` e il programma principale `foto_svd` nel file `svd.f90`, la compilazione del tutto si ottiene col comando

```
g95 interfaccia_svd.f90 svd.f90 -llapack -lblas -o svd
```

che genera il file eseguibile `svd`. Per compilare abbiamo usato il compilatore `g95`. Il file eseguibile `svd` verrà lanciato col comando

```
.\svd
```

Ora vogliamo scrivere un programma che legge i dati dal file `full_svd.txt`, e crea il file `compressa.txt` con i dati della foto compressa.

Per poter salvare in modo efficace i dati compressi nel file `compressa.txt` apriremo il file associandolo ad una unità di scrittura mediante il comando `open(unit=3, file=''compressa.txt'')`, però non è conveniente usare il comando `write(3,*)` per scrivere i dati poiché col formato di scrittura libero denotato da `*` il compilatore distanzierà i valori numerici con molti spazi che ingombrano inutilmente il file. Per forzare il programma a scrivere il file senza spazi aggiuntivi ed inutili occorre utilizzare l'istruzione `write` con un formato opportuno di scrittura. Questo viene fatto nella subroutine `scrivibene` che è riportata di seguito. Useremo i formati `I1, I2, I3, I4` per scrivere un intero di 1,2,3,4 cifre (incluso il segno) senza l'inserimento di spazi aggiuntivi.

```
subroutine scrivibene(unit, x)
  implicit none
  integer :: x, unit
  if(x>=0) then
     if(x<10) then
        write(unit,100)x
     elseif(x<100)then
        write(unit,101)x
     elseif(x<1000)then
        write(unit,102)x
     else
        write(*,*)"ATTENZIONE! intero >= 1000"
     end if
  else
     if(-x<10) then
        write(unit,101)x
     elseif(-x<100)then
        write(unit,102)x
     elseif(-x<1000)then
        write(unit,103)x
     else
        write(*,*)"ATTENZIONE! intero <= -1000"
     end if
  end if
100 format(I1)
101 format(I2)
102 format(I3)
103 format(I4)
```

```
end subroutine scrivibene
```

Il programma che comprime l'immagine legge la svd di $A$ dal file `full_svd.txt`, legge da tastiera il valore di taglio `t` e scrive i dati relativi all'immagine compressa nel file `compressa.txt`. Il programma è allora

```
program comprimi
  implicit none
  integer, parameter :: sp = kind(0.0)
  real(sp), dimension(:,:), allocatable :: u,v
  real(sp), dimension(:), allocatable :: d
  real(sp) :: a, dd, fact
  integer, dimension(:,:), allocatable :: uu, vv
  integer :: m, n, i, j, k, t

  open(unit=2,file="full_svd.txt")
  read(2,*) m, n
  write(*,*)"Foto di dimensioni",m,n
  allocate(u(m,m), v(n,n), d(min(m,n)), uu(m,m), vv(n,n))

  write(*,*) "leggo la svd..."
  do i=1,min(m,n)
     read(2,*) d(i)
  end do
  do i=1,m
     do k=1,min(m,n)
        read(2,*)u(i,k)
     end do
  end do
  do j=1,n
     do k=1,min(m,n)
        read(2,*)v(j,k)
     end do
  end do

  write(*,*)"assegna il valore di taglio (suggerimento 150)"
  read(*,*)t

  do k=1,t
     dd=sqrt(d(k))
     u(:,k)=u(:,k)*dd
     v(:,k)=v(:,k)*dd
  end do
  fact=20
  uu=u*fact
  vv=v*fact
  open(unit=99,file="compressa.txt")
```

```
  write(99,*)m,n,t
  do k=1,t
     do i=1,m
        call scrivibene(99,uu(i,k))
     end do
     do j=1,n
        call scrivibene(99,vv(j,k))
     end do
  end do

end program comprimi
```

Il programma che decomprime l'immagine legge i dati dell'immagine compressa dal file `compressa.txt`, ricostruisce i valori approssimati di $A$ e li salva nel file PGM `decompressa.pgm`. Il programma è allora

```
program decomprimi
  implicit none
  integer, parameter :: sp = kind(0.0)
  real(sp) :: a, dd, fact=20
  integer, dimension(:,:), allocatable :: uu, vv
  integer :: m, n, i, j, k, t, q

  open(unit=2,file="compressa.txt")
  read(2,*) m, n, t
  allocate(uu(m,t), vv(n,t))
  do k=1,t
     do i=1,m
        read(2,*)uu(i,k)
     end do
     do j=1,n
        read(2,*)vv(j,k)
     end do
  end do

  open(unit=3,file="decompressa.pgm")
  write(3,'(A)')'P2'
  write(3,*)n,m
  write(3,*)255
  do i=1,m
     do j=1,n
        a=0
        do k=1,t
           a=a+uu(i,k)*vv(j,k)/fact**2
        end do
        q=a
        q=max(0,q);q=min(q,255)
```

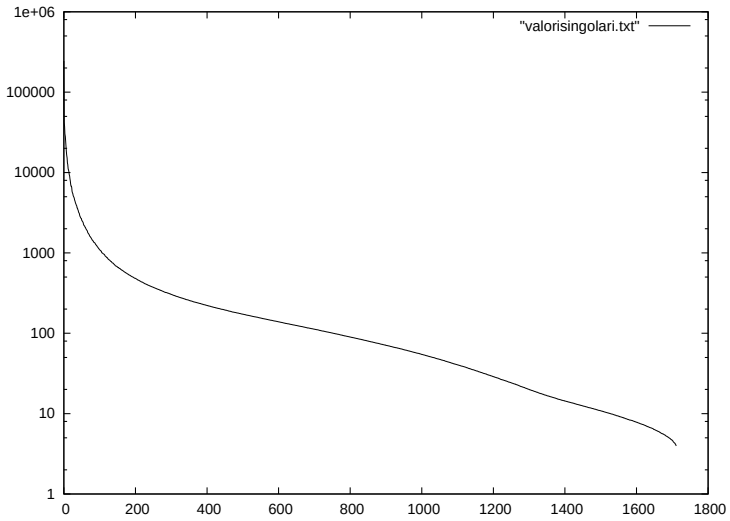


Figura 1: Grafico in scala logaritmica dei valori singolari della foto

```
      call scrivibene(3,q)
    end do
  end do

end program
```

Qui sotto riportiamo i risultati dell'esperimento svolto. Abbiamo considerato la foto $2288 \times 1712$ riportata in figura 2

Abbiamo tracciato il grafico dei valori singolari per evidenziare che questi decadono rapidamente per cui solo la prima parte di essi è significativa al fine di rappresentare correttamente l'immagine. Il grafico è riportato in scala logaritmica nella figura 1. Si osserva che il minimo valore singolare è 3.96 e il massimo è 244405.

Il file compresso con taglio $t = 150$ occupa lo spazio di 1.86 MB a fronte di 14.2 MB dell'immagine originale per un rapporto di compressione di 7.6. L'immagine corrispondente è riportata in figura 3. Il file compresso con taglio $t = 100$ occupa 1.26 MB per un rapporto di compressione di 11.3. L'immagine corrispondente è riportata in figura 4. Il file compresso con taglio $t = 50$ occupa 0.66 MB per un rapporto di compressione di 21. L'immagine corrispondente è riportata in figura 5.

Le immagini possono essere viste usando l'applicazione `display` di linux. Come si può apprezzare dalle immagini riportate, nel taglio con 150 valori singolari non si percepisce alcun degrado della qualità dell'immagine compressa. Qualche leggera alterazione si nota con taglio $t = 100$ mentre con taglio $t = 50$ la qualità dell'immagine compressa diventa scadente.

Figura 2: Foto originale

Figura 3: Compressione con i primi 150 valori singolari, fattore di compressione 7.6.

Figura 4: Compressione con i primi 100 valori singolari, fattore di compressione 11.3.

Figura 5: Compressione con i primi 50 valori singolari, fattore di compressione 21.