Reservoir Computing on the Hypersphere

Zhang Yihang,Scalella Simone

March 9, 2021

Astrazione

Reservoir Computing si riferisce a un framework di reti neurali ricorrenti (RNN), spesso utilizzati per l'apprendimento sequenziale e la previsione. Questo sistema e' costituito da un RNN a peso fisso casuale e un classificatore. Noi ci concentriamo sul problema della sequenza di apprendimento, ed esploriamo un approccio diverso al RC. Più precisamente noi rimuoviamo le funzioni di attivazione neurale non lineare, e consideriamo un reservoir agendo sullo stato di normalizzazione della sfera unitaria. I risultati mostrano che la capacità di memoria del sistema supera la dimensione del reservoir, che è il limite superiore per il tipico approccio RC basato su ECHO STATE NETWORKS. Adesso mostriamo come il sistema proposto può essere applicato a problemi di crittografia simmetrica.

1 Introduzione

Le reti neurali ricorrenti (RCC) sono una classe di metodi di apprendimento automatico, utilizzati per svolgere compiti temporali complessi. La principale caratteristica degli RNN è la loro connettività feed-back, che li trasforma in complessi sistemi di elaborazione delle informazioni in grado di approssimare qualsiasi altro sistema dinamico non lineare con precisione arbitraria [1,2]. Le equazioni dinamiche generiche che descrivono lo stato rnn al tempo t sono:

$$x(t) = f(Vx(t-1) + Us(t)) \tag{1}$$

$$y(t) = g(Wx(t)) \tag{2}$$

dove $s(t) \in \mathbb{R}^M$, $x(t) \in \mathbb{R}^N$ e $y(t) \in \mathbb{R}^K$ sono gli ingressi e le uscite. Le matrici $U \in \mathbb{R}^{N\times M}, V \in \mathbb{R}^{N\times N}$ e $W \in \mathbb{R}^{K\times N}$ sono i pesi di input, output. Le funzioni di attivazione e uscita, f: $\mathbb{R}^N \longrightarrow \mathbb{R}^N$ e g: $\mathbb{R}^N \longrightarrow \mathbb{R}^K$, sono funzioni sigmoidi non lineari, rispettivamente funzioni softmax.
L'addestramento dell'algoritmo implica la regolazione dei pesi, la complessità dell'architettura RNN aumenta la capacità di elaborazione, richiede la "back-propagation" (aggiornamento pesi e baer) degli errori che è computazionalmente costosa,ha convergenza lenta e porta a soluzioni non ottimali [3]. Un approccio diverso, che evita questi problemi è il framework di calcolo del reservoir [4], in questo caso vengono modificati solo i pesi di uscita W, lasciando invariati U e V. Abbiamo il peso fisso casuale (f,V,U) e il classificatore (g,W).
RC contiene diversi modelli di RNN, tra cui la ECHO STATE NETWORK,che è anche la fonte di ispirazione per il modello qui discusso[5], l'ESN richiede la contrattività della funzione di transizione del reservoir (f,U,V), in modo tale che una perturbazione dell'ingresso s(t) sullo stato nascosto x(t) svanisca dopo un certo numero di passi [6]. Una soluzione ampiamente accettata è ridimensionare la matrice del reservoir V in modo tale che il suo raggio spettrale soddisfi $r(v) < 1$ e utilizzare la funzione tanh per attivare gli stati nascosti.
E' dimostrato che la capacità di memoria dell'ESN è limitata dalla dimensionalità della matrice V, $\mu < N$ [8].
Un parametro importante per la capacità della memoria è il raggio spettrale della matrice V, il valore massimo della capacità di memoria si ha quando il raggio spettrale $\rightarrow 1$ [9]. Qui ci concentriamoci sul problema dell'apprendimento, più precisamente rimuovendo la funzione di attivazione neurale non lineare e vincolando la dinamica del sistema all'ipersfera dell'unità utilizzando matrici ortogonali di giacimento, così la capacità di memoria del sistema supera la dimensionalità del recervoir, che è il limite superiore dell' ESN.

2 RC sull'ipersfera

Come discusso nell'introduzione ci concentriamo sul problema dell'apprendimento sequenziale , e modelliamo l'input s(t) come una variabile casuale discreta definita da un insieme M di stati distinti e ortogonali S = 0,1,.....,M - 1,codificato dalle colonne della matrice identità $I = [\delta_i, j]_{M\times M}$ dove:

$$\delta(i,j) = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases} \tag{3}$$

e

$$s(t) = m(t) \Leftrightarrow s(t) = [\delta_{0,m(t)}, ..., \delta_{m(t),m(t)}, ...\delta_{M-1,m(t)}]^T = [0, ..., 1, ...]^T \tag{4}$$

Anche l'output y(t) è una sequenza, la modelliamo come una variabile casuale discreta designata da un insieme K di stati e ortogonali Y = 0,1,...K-1,codificato dalle colonne della matrice identità $I = [\delta_i, j]_{K\times K}$ dove:

$$y(t) = k(t) \Leftrightarrow y(t) = [\delta_{0,k(t)}, ..., \delta_{k(t),k(t)}, ...\delta_{M-1,k(t)}]^T = [0, ..., 1, ...]^T \tag{5}$$

Entrambe soddisfano il vincolo $\|s(t)\| = \|y(t)\| = 1$. Richiediamo che anche la matrice di input di ridimensionamento U $\in \mathbb{R}^{N\times M}$ $(M \leqslant N)$ abbia colonne di lunghezza unitaria, in modo tale che $\|Us(t)\| = 1$ e $\langle Us(t)\rangle = 0, \forall s(t) \in S$.Dobbiamo generare U con gli elementi uniformemente distribuiti in [0,1] e quindi servono 2 fasi di normalizzazione;

$$u_m \leftarrow u_m - \langle u_m\rangle, \tag{6}$$

$$u_m \leftarrow u_m/\|u_m\|, \tag{7}$$

dove u_m, m = 0,1,...,M-1, sono le colonne di U. Ora limitiamo la dinamica dello stato nascosto sull'ipersfera unitaria $\|x(t)\| = 1$ utilizzando matrici di recevoir ortogonali casuali, si possono ottenre per decomposizione QR di qualsiasi matrice RC casuale distribuita normale V = QR, e sostituire V con il fattore ortogonale Q. Poiché Q è ortogonale $Q^TQ = QQ^T = I$, è anche un'isometria, quindi $\|Qx(t)\| = 1 = \|x(t)\|$, questo significa che il raggio spettrale è r(Q) = 1. Ora rimuoviamo la funzione di attivazione non lineare f e consideriamo le seguenti equazioni generiche che descrivono la dinamica del sistema:

$$x(t) = \frac{Qx(t-1) + Us(t)}{\|Qx(t-1) + Us(t)\|}, \tag{8}$$

$$y(t) = g(Wx(t)), \tag{9}$$

con l'uscita g data dalla funzione softmax:

$$y_k(t) = \frac{exp(\langle w_k, x(t)\rangle)}{\sum_{i=0}^{K-1} exp(\langle w_i, x(t)\rangle)}, k = 0, 1, ...K-1, \tag{10}$$

dove w_k è la riga k della matrice W e $\langle .., ..\rangle$ è il prodotto standard. L'apprendimento dei pesi può essere fatto online (iterativamente) o offline (batch). Nell'impostazione originale si raccolgono tutti gli stati nascosti e gli output desiderati come colonne nelle matrici $X = [x(t)]_{N\times T}$ e rispettivamente $Y = [y(t)]_{K\times T}$, t = 0,1,....,T-1, e calcolo la matrice dei pesi di output usando [6,7]:

$$W = YX^\dagger \tag{11}$$

dove

$$X^{\dagger} = \lim_{\eta \to 0_+} X^T (XX^T + \eta I)^{-1} \tag{12}$$

Generalizzazione della matrice inversa al caso in cui non sia quadrata. Per i problemi che richiedono l'apprendimento online è possibile utilizzare metodi iterativi, per esempio si può vedere che la matrice YX^T e XX^T possono essere scritte iterativamente come segue:

$$YX^T(\tau) = \sum_{t=0}^{\tau} y(t)x^T(t), \tag{13}$$

$$XX^T(\tau) = \sum_{t=0}^{\tau} x(t)x^T(t), \tag{14}$$

e quindi si può sempre trovare $W(\tau)$ per qualsiasi passo temporale $0 < \tau < T$. Un metodo alternativo è l'algoritmo "Recursive Least Squares" (RLS) [6,7]. Queste soluzioni però non sono ottimali e hanno problemi di stabilità numerica. Qui preferiamo utilizzare un metodo più semplice basato sulla gradiente discendente, che elimina completamente la necessità di inversione della matrice. Possiamo vedere la matrice dei pesi di output W problema di classificazione con l'insieme $[(x(t), y(t))|y(t) \in Y, t = 0, 1, ..., T-1]$. Pertanto:

$$p(k \equiv y(t)|x(t)) = \frac{exp(\langle w_k, x(t) \rangle)}{\sum_{i=0}^{K-1} x(t) exp(\langle w_i, x(t) \rangle)} \tag{15}$$

Al fine di trovare i pesi, massimizziamo la funzione di verosomiglianza, o allo stesso modo minimizziamo attraverso la funzione di entropia.

$$H(t) = -\sum_{k=0}^{K-1} y_k(t) log p(k|x(t)) \tag{16}$$

Si può facilmente mostrare che il gradiente di H rispetto a wk è:

$$\nabla_{wk} H(t) = (p(k|x(t)) - y_k(t))x^T(t) \tag{17}$$

Pertanto, l'equazione dell'apprendimento online è data da:

$$W(t+1) = W(t) + (y(t) - p(t))x^T(t), \tag{18}$$

dove W(0)= 0 e $p(t) \in \mathbb{R}^{\mathbb{K}}$ è il vettore con le componenti :

$$p_k(t) \equiv p(k|x(t)), k = 0, 1, ..., K-1. \tag{19}$$

3 Regimi associativi vs Generativi

Il sistema RC funziona funziona in un regime associativo, significa che dopo aver imparato è necessario alimentare il sistema con la sequenza s(t) per generare la sequenza di uscita y(t).
Un'altra possibilità interessante è il caso generativo in cui il sistema apprenda una sequenza s(t) ed è in grado di generare la stessa sequenza s(t) a partire dallo stato iniziale s(0) e restituisce ricorsivamente il valore successivo previsto ŝ(t+1):

$$x(t+1) = \frac{Qx(t) + U\hat{s}(t)}{\|Qx(t) + U\hat{s}(t)\|} \tag{20}$$

$$\hat{s}(t+1) = g(Wx(t+1)) \tag{21}$$

In questo caso addestriamo il sistema a partire da x(0) = 0 e sostituisco y(t) con s(t+1), l'equazione diventa:

$$W(t+1) = W(t) + (s(t+1) - p(t+1))x(t+1)^T, \tag{22}$$

dove $p(t+1) \in \mathbb{R}^{\mathbb{M}}$ è il vettore con le componenti:

$$p_m(t+1) \equiv p(m \equiv s(t+1)|x(t+1)), m = 0, 1, ..., M-1. \tag{23}$$

Nel caso offline Y viene sostituito con la matrice $S = [s(t)]_M * T$, t = 0,1,...,T-1, tale che :

$$W = SX^{\dagger} \tag{24}$$

4 Capacità di memoria

Per la memoria del sistema consideriamo il regime generativo e definiamo la capacità di memoria come la massima lunghezza di T, cioè la sequenza che può essere appresa e riprodotta con un errore $0 \leq \epsilon < \theta < 1$, N è la dimensione del resevoir ed M è il numero di possibili statidi ingresso. L'errore della sequenza può essere semplicemente stimato come segue:

$$\epsilon = 1 - T^{(} - 1) \sum_{t=0}^{T-1} \delta(s(t), \hat{s}(t)), \tag{25}$$

dove

$$\delta(a, b) = \begin{cases} 1 & \text{se } a = b \\ 0 & \text{se } a \neq b \end{cases} \tag{26}$$

Ci aspettiamo che la capacità di memoria e l'errore di richiamo ϵ dipendano sia dalla dimensione di N e dal numero di possibili stati di ingresso M.Conviene fissare la lunghezza di T delle sequenze apprese e due quantità $v = N/T \in [0, 1]$ and $\rho = M/T \in [0, 1]$, e per stimare la funzione di errore bidimensionale $\epsilon = \epsilon(v, \rho)$. Possiamo ulteriormente perfezionare il modello considerando l'effetto dell'integrazione leaky sull'attivazione dello stato nascosto, come segue:

$$x(t+1) = \frac{(1-\alpha)x(t) + \alpha(Qx(t) + U\hat{s}(t)}{\|(1-\alpha)x(t) + \alpha(Qx(t) + U\hat{s}(t)\|} \tag{27}$$

$$\hat{s}(t+1) = g(Wx(t+1)), \tag{28}$$

dove $\alpha \in [0, 1]$ è il tasso di integrazione, tale che per $\alpha = 1$ recuperiamo il modello base iniziale. Per questo modello, la capacità di memoria e quindi l'errore di richiamo dipenderanno da tre parametri $\epsilon = \epsilon(v, \rho, \alpha)$.

Una soluzione approssimata per la matrice dei pesi W è ancora possibile per $T > N$, tramite il calcolo della psudo-inversa o del gradiente discendente.
Il ruolo di q è di proiettare in modo ottimale i vettori normalizzati dello stato nascosto x(t) sull'ipersfera, che migliora il condizionamento della matrice X, e quindi la stabilità di W.
Il classificatore softmax calcola l'indice corrispondente alla coordinata massima di s(t+1) = g(Wx(t+1)). Quindi un vettore di output approssimativo s(t+1), ottenuto proiettando lo stato nascosto x(t+1) con una matrice approssimativa W, dovrebbe essere sufficiente per lo scopo di classificazione.
L'aumento di capacità di memoria del modello RC è una conseguenza del rilassamento della soluzione.
Un parametro che influenza la capacità di memoria è il numero di possibili stati distinti di input (ortogonali) M, da risultati emerge che se aumento M si riduce l'errore e aumenta la memoria, ma è strano,poiché ci si aspetta che sia più difficile imparare sequenze complesse rispetto a quelle semplici. Tuttavia aumentando M aumentiamo la dimensione di $M \times T$ della matrice $S \in \mathbb{R}^{\mathbb{M}\times\mathbb{T}}$ e anche la dimensione della matrice dei pesi $W = SX^{\dagger} \in \mathbb{R}^{\mathbb{M}\times\mathbb{N}}$ e quindi aumentiamo il "supporto fisico" della memoria. Quindi per avere un aumento della capacità di memoria aumentiamo M.
Abbiamo anche notato che il comportamento del sistema è indipendente dal "tipo" di matrice ortogonale utilizzato per il reservoir. Con il metodo QR si ottiene una matrice ortogonale Q piena e densa. Tuttavia, è possibile usare matrici di permutazione, che sono anche ortogonali, e il sistema mantiene la sua capacità di memoria. In particolare, le matrici di permutazione cicliche sono molto utili poiché la loro applicazione allo stato nascosto x(t) consiste semplicemente nello spostare tutti gli elementi di un passo in avanti e inserire il primo elemento sul retro. Pertanto, la mappatura della permutazione ciclica può essere

implementata semplicemente come segue:

$$x_n(t) \longleftarrow x_{(n+1)modN}(t), n = 0, 1, ...N - 1, \tag{29}$$

ed elimina completamente la necessità di immagazzinamento e accelera notevolmente il calcolo da $O(N^2)$ a $O(N)$.

5 Applicazione nella crittografia simmetrica

Recentemente è stato suggerito che gli ESN potrebbero essere utilizzati anche in crittografia [10]. In questo contesto, Alice e Bob si scambiano messaggi e cercano di proteggere le loro comunicazioni dalle intercettazioni di Eve. Alla fine per scambiarsi i messaggi, sia Alice che Bob condividono una copia identica di un ESN. Per crittografare un messaggio, Alice addestra l'ESN in modo che l'ESN riproduca l'input. Quindi invia i pesi di uscita W a Bob, Bob li usa per decrittografare il messaggio. Senza la corrispondente struttura interna ESN (U,V) Eve non sarà in grado di decifrare il messaggio. Qui, ovviamente, la struttura interna dell'ESN (U,V) è la chiave segreta.
L'approccio di cui sopra, basato sul regime generativo del RC (o ESN), non è l'unica soluzione. Qui descriviamo un metodo diverso basato sul regime associativo, che aggiunge anche diverse sicurezza extra. Dal momento che ipotizziamo che l' RC operi in regime associativo, oltre che la struttura interna dell'RC possiamo anche usare la stringa di input s(t) come chiave segreta (U,Q,s(t)), e considerare che la la stringa di output y(t) è il messaggio da crittografare.Per aumentare la sicurezza utilizziamo anche matrici piene,dense e ortogonali Q di reservoir.In uno scenario del genere, sia Alice che Bob hanno una copia dell' RC. Alice usa l'RC per crittografare un messaggio y(t) con la chiave segreta (U,Q,s(t)), quindi invia i pesi di output W a Bob che li usa insieme alla stessa chiave segreta (U,Q,s(t)) per decrittografare il messaggio.Questo è anche molto conveniente perché la stringa di input segreta s(t) può essere generata casualmente da un file stringa segreta (password), ad esempio utilizzando una funzione hash unidirezionale sicura, tale che blocchi di testo con la lunghezza uguale all'output della funzione hash possono essere crittografati e decrittografati. Inoltre,prima della fase di apprendimento dell' RC, come elemento di ulteriore sicurezza, il messaggio può essere crittografato con un semplice passaggio XOR, utilizzando la stessa chiave segreta $s(t) : y(t) \longleftarrow y(t) \otimes s(t)$. Ovviamente, la decrittazione richiede anche l'applicazione di questo passaggio aggiuntivo. La fase di pre-crittografia e post-decrittografia XOR può essere utilizzata anche per la generazione a regime, rendendolo più resistente agli attacchi.

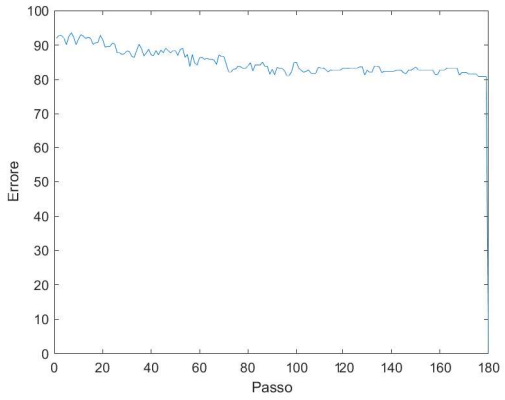

Figure 1: Questo è l'andamento dell'errore in funzione dei valori del codice Matlab riportato di seguito

6 Conclusioni

In questo articolo abbiamo presentato un nuovo modello RC con le dinamiche vincolate all'ipersfera unitaria.Il modello rimuove la funzione di attivazione neu-

rale non lineare e utilizza matrici di reservoir ortogonali.I nostri risultati numerici hanno mostrato che la capacità di memoria del sistema è maggiore della dimensionalità del reservoir, che è il limite superiore per il tipico approccio RC basato su ESN. Abbiamo anche discusso l'applicazione del sistema alla crittografia simmetrica, e abbiamo dato diversi suggerimenti per migliorare la sua robustezza e sicurezza. Nell'Appendice è stata inclusa anche un'implementazione numerica.

7 Appendice

Qui diamo un'implementazione Python (Numpy) per il caso con permutazioni cicliche, che è anche il più veloce. Il programma apprende il primo paragrafo da "Le avventure di Sherlock Holmes" di Sir Arthur Conan Doyle. Il testo ha T = 1137 caratteri, con M = 38 caratteri distinti. Con i parametri del programma, $N = T/2$ e $\alpha = 0,5$, il metodo di apprendimento online converge in 291 iterazioni.

```
 Reservoir computing on the hypersphere
import numpy as np
def init(M,N):
u,v = np.random.rand(N,M),np.identity(M)
for m in range(M):
u[:,m] = u[:,m] - u[:,m].mean()
u[:,m] = u[:,m]/np.linalg.norm(u[:,m])
return u,v
def recall(T,N,w,u,c,a,ss):
x,i = np.zeros(N),ci[ss]
for t in range(T-1):
10
x = (1.0-a)*x + a*(u[:,i] + np.roll(x,1))
x = x/np.linalg.norm(x)
y = np.exp(np.dot(w,x))
i = np.argmax(y/np.sum(y))
ss = ss + str(c[i])
return ss
def error(s,ss):
err = 0.
for t in range(len(s)):
err = err + (s[t]!=ss[t])
return np.round(err*100.0/len(s),2)
def offline_learning(u,v,c,a,s):
T,(N,M),eta = len(s),u.shape,1e-7
X,S,x = np.zeros((N,T-1)),np.zeros((M,T-1)),np.zeros(N)
for t in range(T-1):
x = (1.0-a)*x + a*(u[:,ci[s[t]]] + np.roll(x,1))
x = x/np.linalg.norm(x)
```

```
X[:,t],S[:,t] = x,v[:,ci[s[t+1]]]
XX = np.dot(X,X.T)
for n in range(N):
XX[n,n] = XX[n,n] + eta
w = np.dot(np.dot(S,X.T),np.linalg.inv(XX))
ss = recall(T,N,w,u,c,alpha,s[0])
print "err=",error(s,ss),"%\n",ss,"\n"
return ss,w
def online_learning(u,v,c,a,s):
T,(N,M) = len(s),u.shape
w,err,tt = np.zeros((M,N)),100.,0
while err>0 and tt<T:
x = np.zeros(N)
for t in range(T-1):
x = (1.0-a)*x + a*(u[:,ci[s[t]]] + np.roll(x,1))
x = x/np.linalg.norm(x)
p = np.exp(np.dot(w,x))
p = p/np.sum(p)
w = w + np.outer(v[:,ci[s[t+1]]]-p,x)
ss = recall(T,N,w,u,c,a,s[0])
err,tt = error(s,ss),tt+1
print tt,"err=",err,"%\n",ss,"\n"
return ss,w
s = \
"To Sherlock Holmes she is always THE woman. I have seldom heard ...
    him mention \
her any other name. In his eyes she eclipses and predominates ...
    the whole of \
her sex. It was not that he felt any emotion akin to love for ...
    Irene Adler. \
All emotions, and that one particularly, were abhorrent to his ...
    cold, precise \
but admirably balanced mind. He was, I take it, the most perfect ...
    reasoning \
and observing machine that the world has seen, but as a lover he ...
    would have \
placed himself in a false position. He never spoke of the softer ...
    passions, \
save with a gibe and a sneer. They were admirable things for the ...
    observer--\
excellent for drawing the veil from men's motives and actions. ...
    But for the \
trained reasoner to admit such intrusions into his own delicate ...
    and finely \
adjusted temperament was to introduce a distracting factor which ...
    might throw \
a doubt upon all his mental results. Grit in a sensitive ...
    instrument, or a crack \
in one of his own high-power lenses, would not be more ...
    disturbing than a strong \
emotion in a nature such as his. And yet there was but one woman ...
    to him, and \
that woman was the late Irene Adler, of dubious and questionable ...
    memory."

c = list(set(s))
ci = {ch:m for m,ch in enumerate(c)}
```

```
T,M = len(s),len(c)
N,alpha = int(0.5*T),0.5
np.random.seed(12345)
u,v = init(M,N)
ss,w = offline_learning(u,v,c,alpha,s)
ss,w = online_learning(u,v,c,alpha,s)
print T,N,M,alpha
```

8 Implementazione in matlab

Programma per il test del metodo "Reservoir computing on the hypersphere (Andrecut (2017)" di Simone Scalella e Yihang Zhang (CdS - Informatica e Automazione, UnivPM) Rivisto da S. Fiori (DII, UnivPM) - Ottobre 2020

```
clc; clear; close all;
% Stringa di prova
s = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. ...
    Suspendisse mi tellus, tempus id dictum et, pharetra et ...
    lorem."+...
    "Praesent nulla purus, porttitor nec ligula at, rutrum ...
        hendrerit leo. Nunc quis metus urna. Praesent nec ...
        pellentesque dui."+...
    "Donec blandit libero ac lacus commodo vulputate. Etiam ut ...
        dapibus dui, a scelerisque libero. Morbi convallis ...
        euismod nulla,"+...
    "eget dignissim magna mollis sed. Fusce sagittis at arcu ...
        fringilla sagittis. Maecenas dolor turpis, pellentesque ...
        vitae nulla eu,"+...
    "fringilla congue lorem. Vivamus feugiat placerat eros non ...
        vestibulum. Nunc consequat sed lectus sit amet commodo."+...
    "Curabitur eget risus auctor, congue mauris in, consectetur ...
        ex. Pellentesque dapibus at turpis commodo facilisis.";
    "Nam turpis mi, condimentum sed enim vel, condimentum ...
        placerat eros. Nullam placerat lorem magna, eget feugiat ...
        ante dictum sed."+...
    "Phasellus efficitur hendrerit nisl non interdum. Aenean ...
        molestie mi felis, ullamcorper venenatis dui pulvinar ...
        et. "+...
    "Praesent rhoncus at mi in pretium. Aenean nec felis libero. ...
        Nullam pharetra ullamcorper neque, ut aliquam ex ...
        volutpat eget."+...
    "Aenean facilisis non sapien a ultricies. Cras sit amet ex ...
        eget sapien ullamcorper commodo. Nulla sit amet est ...
        neque. Ut nec imperdiet tortor."+...
    "In pulvinar quam est, non pellentesque urna dapibus in. ...
        Aenean felis nisi, convallis sed congue ac, pulvinar ut ...
        erat."; % Quisque quis neque quis velit sollicitudin ...
        pharetra sit amet tincidunt quam. Donec condimentum dui ...
        non risus tempor scelerisque. Praesent et tristique ...
```

```
          nulla. Vestibulum lacinia diam risus, sollicitudin ...
          mollis urna aliquet quis. Phasellus eu accumsan nisi. ...
          Aliquam erat volutpat. Curabitur et ligula non dui ...
          pharetra cursus. Pellentesque in porttitor velit. Aenean ...
          molestie dui non consequat aliquam. Proin viverra purus ...
          metus, vel suscipit metus cursus a. Quisque luctus ...
          consequat ante, in ullamcorper urna imperdiet ut. Nunc ...
          quis leo pharetra, lacinia erat vitae, porttitor sem. ...
          Mauris vel aliquet risus. Vestibulum mattis convallis ...
          urna, in lobortis magna sodales nec. Vestibulum mollis ...
          ex sit amet volutpat eleifend. Phasellus interdum ...
          fermentum augue. Sed malesuada ipsum vitae purus mattis ...
          elementum. Orci varius natoque penatibus et magnis dis ...
          parturient montes, nascetur ridiculus mus. Sed aliquet ...
          varius auctor. Nulla dignissim ex eu sapien eleifend, ...
          efficitur auctor ante malesuada. Duis sed lectus in ...
          justo feugiat tristique eu sed enim.";

s1 = split(s,""); s1(1) = []; s1(numel(s1))=[]; c = unique(s1);
T = numel(s1); M = numel(c);
% rp = randperm(M); ci = c(rp); c = ci; %% Secondo me non serve
N = round(0.5*T); alpha = 0.5;
rng(12345);
[u,v] = init(N,M);
[ss,w,errh] = online_learning(u,v,c,alpha,s1,c);
plot(errh);xlabel('Passo');ylabel('Errore');
%[ss,w] = offline_learning(u,v,c,alpha,s1,ci);

function [u,v] = init(N,M)
    u = rand(N,M);
    v = eye(M);
    for i = 1:M
        u(:,i) = u(:,i)-mean(u(:,i));
        u(:,i) = u(:,i)/norm(u(:,i),'fro');
    end
end

function ss = recall(T,N,w,u,c,a,ss1,ci)
    x = zeros(N,1);
    i = find(ci==ss1);
    ss =ss1;
    for j = 1:T-1
        x = (1.0-a)*x+a*(u(:,i)+circshift(x,1));
        x = x/norm(x,'fro');
        y = exp(w*x);
        [¬,i] = max(y/sum(y));
        ss = [ss; c(i)];
    end
end

function erro = error(s,ss)
    err = 0;
    for i = 1:numel(s)
        err = err+(s(i) ≠ ss(i));
    end
    erro = round(err*100/numel(s),2);
end
```

```
54
55 function [ss,w,errh] = online_learning(u,v,c,a,s,ci)
56  T = numel(s);
57  [N,M] = size(u);
58  w = zeros(M,N);
59  err = 100;
60  tt = 1;
61  errh = [];
62 while(err>0 && tt≤T)
63     x = zeros(N,1);
64     for i = 1:T-1
65         x = (1.0-a)*x+a*(u(:,find(ci==s(i)))+circshift(x,1));
66         x = x/norm(x,'fro');
67         p = exp(w*x);
68         p = p/sum(p,'all');
69         w = w+(v(:,find(ci==s(i+1),1))-p)*x';
70     end
71     ss = recall(T,N,w,u,c,a,s(1),ci);
72     err = error(s,ss)
73     tt = tt+1;
74     errh = [errh err];
75  end
76 end
77
78 % % function [ss,w] = offline_learning(u,v,c,a,s,ci)
79 % %  T = numel(s);
80 % %  [N,M] = size(u);
81 % %  eta = 1e-7;
82 % %  X = zeros(N,T);
83 % %  S = zeros(M,T);
84 % %  x = zeros(N,1);
85 % %  for i=1:T-1
86 % %     %x1 = (1.0-a)*x %DGN
87 % %     %x2 = a*(u(:,find(ci==s(i))).'+circshift(x,1))%DGN
88 % %     x = (1.0-a)*x+a*(u(:,find(ci==s(i)))+circshift(x,1));
89 % %     x = x/norm(x,'fro');
90 % %     X(:,i) = x;
91 % %     S(:,i) = v(:,find(ci==s(i+1),1));
92 % %  end
93 % %  XX = X*X.';
94 % %  for j = 1:N
95 % %     XX(j,j) = XX(j,j)+eta;
96 % %  end
97 % %     w = S*X.'*inv(XX);
98 % %     %[offlinew1 offlinew2] = size(w)%DGN
99 % %     %[su1offline su2offline ] = size(u) %DGN
100 % %     ss = recall(T,N,w,u,c,a,s(1),ci);
101 % %     %n_ss = numel(ss) %DGN
102 % %     "err = "
103 % %     error(s,ss)
104 % %     %fprintf("err = ",error(s,ss))
105 % %
106 % % end
107 % %
```