Week 2 — Assignment Submission

Gianluca Scarpellini - 807541 - g.scarpellini1[at]disco.unimib.it

27 ottobre 2019

Indice

1 Task

Il task affrontato prevedeva una duplice obiettivo: l'addestramneto di un classificatore e l'apprendimento tramite autoencoder della distribuzione di probabilità dello spazio di input. Abbiamo pertanto deciso di effettuare esperimenti con differenti tipi di approcci. Abbiamo innanzitutto sviluppato un'architettura 'Fully Connected' [3] al fine di effettuare un task di classificazione. Successivamente, in modo del tutto indipendente, abbiamo sviluppato un'autoencoder con cui apprendere una compressione in uno spazio delle feature ridotto della distribuzione di input e il relativo decoder di decompressione. Abbiamo quindi ipotizzato, seguendo le indicazioni circa la capacità di rappresentazione di una rete neurale [GBC16], che la somma delle parti potesse dare risultati più soddisfacenti dei singoli approcci. Abbiamo pertanto effettuato diversi esperimenti con un autoencoder dotato di un output di classificazione, addestrando lo stesso a effettaure sia il task di classificazione sia il task di ricostruzione. Per rendere l'autoencoder più robusto abbiamo sviluppato layer ad-hoc che introducessero rumore nell'input (Salt-And-Pepper) e regolarizzazioni che producessero

sparsità nei pesi appresi. Abbiamo infine implementato e testato autencoders simmetrici i cui encoder e decoder condividessero la stessa matrice dei pesi, come suggerito in [Gér17]. Le nostre deduzioni si sono rivelate corrette. Nella sezione esperimenti [4] riportiamo risultati ottenuti nei due task, dandone una giustificazione.

2 Dataset e preprocessing

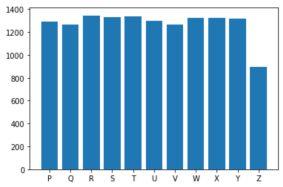


Figura 1: Distribuzione del dataset

Il dataset del task in oggetto [**??**] è una versione alfabetica del più noto MINST [LC10]. Si tratta di un set di 14,000 immagini dei caratteri dell'alfabeto dalla lettera P alla lettera Z. Le classi associate a ciascun carattere sono relative alla posizione della lettera nell'alfabeto (ossia, da 16 a 26 inclusi); abbiamo quindi deciso di traslarle le stesse riportandoci a un range di valori [0:11]. Al fine di attenerci alla consegna, che prevedeva l'utilizzo di reti neurali fully connected e quindi privi di layer convoluzionali, abbiamo effettuato il flat delle immagini ottenendo vettori di shape (784,). Abbiamo quindi convertito i vettori, in scala 0:255, in vettori di valori compresi tra 0 e 1. Il processo di rescaling delle intensità riduce la varianza del dataset e rende i dati più facilmente fruibili per una rete.

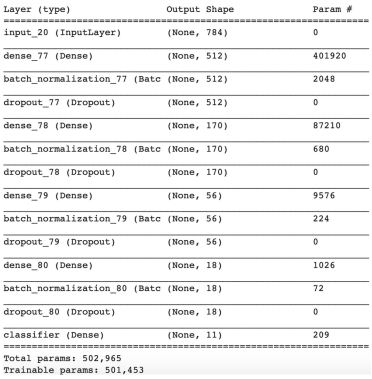
Figura 2: Estratto del dataset

3 Modelli

3.1 Rete neurale

Abbiamo deciso di effettuare una prima serie di esperimenti impiegando una rete neurale fully connected. La rete possiede 500,000 parametri così suddivisi:

- 5 Layer densi di dimensione 512, 170, 56, 18, 11. Dopo aver deciso la dimensione iniziale (512) e finale (32) degli hidden layers, abbiamo diviso la dimensione per 2 al fine di ottenere layer densi con depth multipla di 2, che permette di sfruttare le ottimizzazioni delle moderne GPU. Ciascun layer viene inizializzato con he_normal, che ben si adatta alla funzione di attivazione relu. Abbiamo optato per un'attivazione non lineare per la sua robustezza e per ridurre il fenomeno di clipping del gradiente durante il training.
- 4 Layers di dropout, per introdurre rumore nel processo di apprendimento
- 4 Layers di Batch Normalization, che garantisce che gli output dei layer densi siano riscalati e normalizzati. La normalizzazione garantisce che il layer successivo riceva vettori distribuiti con una gaussiana, minimizzando la varianza.



Layer (type)	Output Shape	Param #
input_20 (InputLayer)	(None, 784)	0
dense_77 (Dense)	(None, 512)	401920
batch_normalization_77 (Batc	(None, 512)	2048
dropout_77 (Dropout)	(None, 512)	0
dense_78 (Dense)	(None, 170)	87210
batch_normalization_78 (Batc	(None, 170)	680
dropout_78 (Dropout)	(None, 170)	0
dense_79 (Dense)	(None, 56)	9576
batch_normalization_79 (Batc	(None, 56)	224
dropout_79 (Dropout)	(None, 56)	0
dense_80 (Dense)	(None, 18)	1026
batch_normalization_80 (Batc	(None, 18)	72
dropout_80 (Dropout)	(None, 18)	0
classifier (Dense)	(None, 11)	209

Total params: 502,965
Trainable params: 501,453

Figura 3: Rete Neurale Fully Connected impiegata

3.2 Autoencoders e tying autoencoders

Abbiamo effettuato il task di ricostruzione sfruttando un approccio a Autoencoders. I modelli che abbiamo deciso di impiegare, con diverso numero di layer e numero di neuroni, hanno seguito un template simmetrico. Di conseguenza, seguendo le indicazion di [Gér17], abbiamo deciso di impiegare le stesse matrici dei pesi sia in fase di encoding sia in fase di decoding (traposte [1]).

$$W_{N-L+1} = W_L^T \tag{1}$$

Tale approccio, detto **tying autencoders**, ci ha permesso di dimezzare il numero di parametri in fase di training e ridurre il rischio di overfitting. In 4 riportiamo un confronto tra ricostruzioni che dimostra visualmente la bontà dell'approccio.

Figura 4: Ricostruzione con autoencoder

4 Esperimenti

Riportiamo di seguito una tabella di alcuni esperimenti effettuati. Per tutti gli esperimenti abbiamo deicso di mantenere lo stesso algoritmo di ottimizzazione, Adam. Adam difatti è un ottimizzatore con learning rate adattivo, che si dimostra spesso in letteratura un approccio ottimale. Abbiamo effettuato esperimenti anche con ottimizzatore RMSProp, senza trarne vantaggio. La scelta ha ridotto quindi il numero di iperparametri da scegliere: abbiamo optato per mantenere il learning rate a 0.001 e batch size 128, in quanto entrambi non sembravano incidere sui risultati. Diverse dimensioni di batch size sono invece state provate per l'addestramento degli autoencoders. Per il task di ricostruzione, ossia di regressione dei valori dei pixel, abbiamo impiegato MSE come loss; per il task di classificazione multi-classe abbiamo invece utilizzato categorical_crossentropy.

$$MSE(\hat{y}, y) = \frac{\sum_i^n (\hat{y}_i - y_i)^2}{n} \tag{2}$$

4.1 Classificazione

Abbiamo ottenuto risultati equiparabili allo stato dell'arte in dataset similari quali il MNIST. In particolare, entrmabi i tre approcci impiegati per classificazione (Fully Connected Neural Network, Auto Encoder con output multipli e training distinto di AE e classificatore concatendato all'Encoder) hanno raggiunto f1-score di *0.94*. Riportiamo in coda una tabella riassuntiva degli esperimenti effettuati [10].

4.2 Ricostruzione

Riportiamo diverse ricostruzioni ottenute con i tre approcci: autoencoders, autoncoders con classificatore e tying-autoencoders. Abbiamo ottenuto valor di mse rispettivamente di *0.242 0.292* e *0.019* su validation. Dal confronto notiamo la bontà di ricostruzione dei tre approcci [4,5, 6] ci sentiamo affermare che la ricostruzione del primo è più fedele rispetto agli altri due. Abbiamo inoltre generato la rappresentazione bidimensionale del manifold ottenuto con i tre approcci [7, 8, 9].

Figura 5: Ricostruzione con tying autoencoder

Figura 6: Ricostruzione con autoencoder multiple-output

5 Conclusione

In conclusione, abbiamo valutato diversi approcci di classificazione e di ricostruzione a partire del dataset nostra disposizione. Abbiamo confrontato visivamente e con MSE le ricostruzioni ottenute [5, 4, 6] e proiettato visivamente i manifold che abbiamo ottenuto [8, 9, 7], riscontrando la bontà dell'approccio tying autoencoders. Abbiamo inoltre impiegato diversi approcci per il task di classficazione, ottenendo risultati tra loro equivalenti.

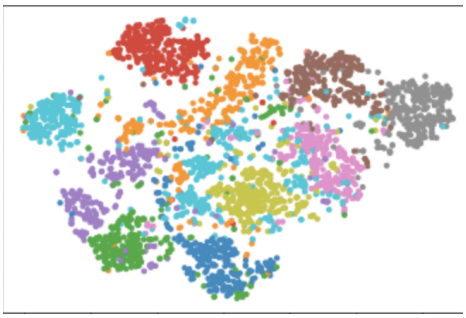


Figura 7: Manifold ottenuto con autoencoder multiple-output

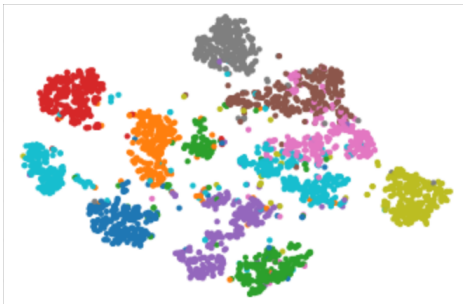


Figura 8: Manifold ottenuto con autoencoder

Figura 9: Manifold ottenuto con autoencoder multiple-output

exps	AE as feature extractor	AE regulizer	epochs	standard	batch size	CV	early stopping	lr decay on plateau	Noise	f1score
1	FALSO	FALSO	300	VERO	128	5	10	FALSO	False	**0.93**
2	FALSO	FALSO	300	FALSO	128	5	10	FALSO	False	**0.94**
3	FALSO	FALSO	300	FALSO	128	5	10	0.1, patience 10	False	**0.94**
4	FALSO	FALSO	300	FALSO	128	5	10	FALSO	Gaussian	**0.94**
5	FALSO	FALSO	300	FALSO	128	5	10	FALSO	saltpepper	**0.93**
6	VERO	FALSO	300	FALSO	128	FALSO	10	FALSO	FALSE	**0.91**
7	VERO	FALSO	300	FALSO	128	FALSO	10	FALSO	saltpepper	**0.93**
8	VERO	FALSO	300	FALSO	128	FALSO	10	FALSO	gaussian	**0.94**
9	VERO	KLD	300	FALSO	128	FALSO	10	FALSO	FALSO	**0.92**
10	VERO	KLD	300	FALSO	256	FALSO	10	FALSO	Gaussian	**0.94**

Figura 10: Tabella riassuntiva degli esperimenti effettuati

Bibliografia

[LC10] Yann LeCun e Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[GBC16] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Gér17] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1491962299.