Studio e predizione di espressioni facciali per modelli 3D

Alessandro Sestini
Matricola - 6226094
alessandro.sestini@stud.unifi.it

Francesco Lombardi
Matricola - 6344416
francesco.lombardi3@stud.unifi.it

Abstract

Per questo elaborato abbiamo effettuato lo studio e l'analisi di una collezione contenente espressioni facciali, con lo scopo di costruire un predittore di trasformazioni efficiente e credibile da usare per realizzare una sintesi di facce espressive; questo è stato fatto per cercare di popolare un eventuale dataset da usare per addestrare delle reti neurali, qualora avessero pochi esempi a disposizione. Il lavoro si è svolto in più parti: come primo passo abbiamo studiato il significato dei dati ed effettuato delle prove sul funzionamento degli script di modellazione delle facce; in seguito abbiamo creato due tipi di predittori: basati sui soli concetti statistici di media, mediana e moda oppure regressori basati su tecniche di machine learning; infine abbiamo testato il tutto manipolando le facce neutre e valutando qualitativamente i risultati ottenuti. Come vedremo nel corso della relazione le facce ottenute sono risultate essere abbastanza credibili con qualunque tipo di tecnica studiata, con il regressore SVR leggermente in vantaggio rispetto alle altre.

1. Introduzione

Allo stato attuale, uno dei più grandi problemi da affrontare nell'uso delle reti neurali è l'apprendimento: per funzionare correttamente questi modelli necessitano di una grande quantità di *training data*, che in certi contesti potrebbe essere difficile da reperire. In particolare, il nostro elaborato si concentra sull'ambito delle espressioni facciali, per il quale l'operazione di raccolta immagini di diverse espressioni per lo stesso soggetto potrebbe risultare notevolmente complessa: il problema principale risiede nel cercare un elevato numero di persone disponibili ad essere fotografate e che siano disposte ad effettuare diverse espressioni. Dunque ci siamo concentrati sul trovare un metodo per fare sintesi di espressioni partendo da una semplice faccia neutra, in modo da diminuire il numero di dati necessari e allo stesso tempo di aumentare le dimensioni del dataset da passare in input alla rete.

Il primo passo è stato quindi l'analisi dei dati a nostra disposizione: la collezione è composta da una matrice di circa 600 vettori colonna, ognuno dei quali ha 300 componenti; i campioni possono descrivere sia la rappresentazione di una vera faccia neutra da applicare ad un modello medio per ottenere il campione 3D della persona (eventualmente con la relativa texture), sia la trasformazione necessaria per portare la mesh già modellata nell'espressione fotografata per l'occasione. Le colonne rappresentano alternativamente un vettore per faccia neutra e il vettore per la trasformazione espressiva della precedente. In figura 1 è mostrato un esempio:

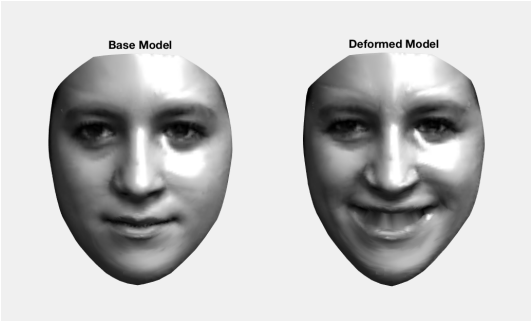


Figura 1: Esempio di espressione felice presente nel dataset

La matrice e gli script per la realizzazione dei modelli 3D sono stati implementati precedentemente a questo progetto in *MATLAB*[5], mentre per il nostro lavoro abbiamo preferito utilizzare il linguaggio *Python* in quanto ci ha permesso di sviluppare approcci statistici e di machine learning in maniera più semplice e diretta: si è reso quindi necessario l'uso delle *MATLAB Engine API*[4] che fornisce un pacchetto per *Python* atto a richiamare gli script per la modellazione e la visualizzazione delle facce direttamente da questo linguaggio.

2. Statistica

La prima baseline adottata è stata dunque quella di studiare la statistica dei dati a nostra disposizione, nel tentativo di ottenere in maniera più semplice possibile un vettore

generale descrittivo di una particolare espressione tramite i concetti di media, mediana e moda. Nella tabella 1 è possibile vedere il numero di vettori presente per espressione:

Espressione	Numero di vettori
happy	69
angry	45
disgust	59
fear	25
contempt	18
surprise	83
sadness	28

Tabella 1: Numero di vettori per espressione

Alcuni esempi sono però sbagliati: nel dataset sono presenti degli array etichettati "neutral" che in realtà non sono facce neutre, e dei vettori etichettati come espressivi quando invece sono neutri oppure di espressioni diverse da quelle evidenziate.

Il calcolo della media e della mediana è stato fatto tramite la libreria *numpy*[2] per Python, mentre per il calcolo della moda abbiamo usato la libreria *sklearn* [6]; le facce ottenute tramite questi metodi sono visibili nella sezione dei risultati.

2.1. Media

La media serve a descrivere tramite un solo vettore l'insieme dei dati per la singola espressione. Per calcolarla, abbiamo semplicemente utilizzato la media aritmetica:

$$m(i) = \frac{1}{n}\sum_{j=1}^{n} x_j(i)$$

dove $m(i)$ è la componente i-esima del vettore media e $x_j(i)$ è la componente i-esima del vettore trasformazione j-esimo.

2.2. Mediana

Si definisce mediana il valore assunto dalle unità statistiche che si trovano nel mezzo della distribuzione. Viene generalmente preferita alla media quando sono presenti degli outlier che non portano informazioni ma che spostano il valor medio verso direzioni fuorvianti. Per calcolarla si ordinano i valori delle stesse componenti dei vettori delle singole espressioni e si prende quello che si pone al centro di questo ordinamento, costruendo così un nuovo array di 300 elementi, definiti dai mediani, descrittivo della particolare espressione.

2.3. Moda

La moda è semplicemente il punto in cui troviamo la più grande concentrazione dei dati. Per calcolarla abbiamo usato l'algoritmo Mean Shift, una procedura in grado di localizzare i massimi di una funzione di densità dati dei campioni di quella funzione. È un algoritmo iterativo basato sulla ricerca di centroidi, che lavora aggiornando questi punti notevoli in modo che siano rappresentativi della media di una particolare regione dello spazio dei dati. Supponiamo di avere degli x_i centroidi iniziali e un kernel $K(x_i - x)$, che determina il peso dei punti vicini per la stima della media; tipicamente viene usato un kernel gaussiano:

$$K(x_i - x) = e^{-c||x_i - x||^2}$$

Dato un centroide x_i all'iterazione t, questo viene aggiornato all'iterazione $t+1$ tramite la formula:

$$x_i^{t+1} = x_i^t + m(x)$$

con

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i)x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

dove $N(x_i)$ è un intorno di x_i, ovvero l'insieme dei punti in cui $K(x_i) \neq 0$ e $m(x)$ viene chiamato *vettore di mean shift*; l'aggiornamento dei centroidi viene ripetuto fino alla convergenza di $m(x)$. In figura 2 è mostrato un esempio di ricerca delle mode in uno spazio 2D:

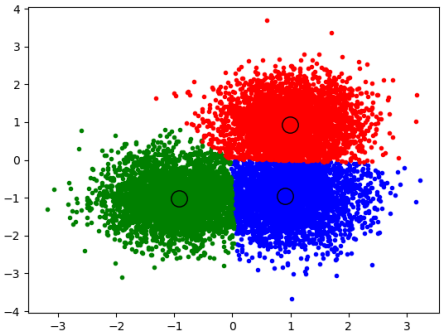


Figura 2: Esempio di applicazione dell'algoritmo mean shift su uno spazio di campioni 2D

L'unico parametro che prende in ingresso questo algoritmo è il raggio della regione gaussiana, tipicamente indicata con *bandwidth*. Il nostro obiettivo era quello di trovare il centroide che rappresentasse al meglio la disposizione dei dati, quindi il punto massimo della funzione di distribuzione: per fare questo, era necessario che l'algoritmo ci restituisse un solo punto. Siamo quindi partiti mettendo una *bandwidth* fissa e abbiamo ripetuto il mean shift aumentando leggermente il raggio fino a che la procedura non ci avesse restituito un solo punto; abbiamo quindi preso questo centroide come vettore descrittivo della nostra distribuzione riguardante le singole espressioni.

2.4. Studio della distribuzione tramite Mean Shift

L'algoritmo Mean Shift è risultato utile anche per studiare più approfonditamente la disposizione e il significato dei dati. Il primo esperimento che è stato fatto è stato quello di fissare la *bandwidth*, applicare l'algoritmo, prendere i centroidi risultanti e contare quanti campioni cadevano nella regione di influenza di tutti i centroidi. Nella tabella 2 sono riportati i risultati per singola espressione; la *bandwidth* usata è stata trovata tramite il metodo *estimate_bandwidth* di *sklearn*.

Espressione	# Centroidi	Percentuale
happy	5	92%
angry	4	93%
disgust	2	98%
fear	2	96%
contempt	5	73%
surprise	2	95%
sadness	3	93%

Tabella 2: La tabella mostra il numero di centroidi trovati fissando la *bandwidth* e la percentuale di vettori che cadono all'interno della regione del primo centroide

Da questo test è facile capire quanto il primo centroide, quello che si trova sul picco massimo della distribuzione, sia quello più rappresentativo dei campioni: probabilmente gli altri massimi vanno a catturare i possibili outlier o gli errori presenti nel dataset. L'altra sperimentazione fatta è stata quella di iterare l'algoritmo aumentando leggermente la *bandwidth* fino a trovare 2 centroidi. Abbiamo quindi confrontato le facce ottenute applicando entrambi i vettori descritti dai due massimi per ogni espressione: dal primo estraevamo la stessa faccia che ottenevamo utilizzando la precedente strategia in cui avevamo un solo centroide; dal secondo vettore estraevamo invece una faccia non credibile. In figura 3 è mostrato il risultato per l'espressione "contempt", quella risultata più critica (vedi la tabella 2)

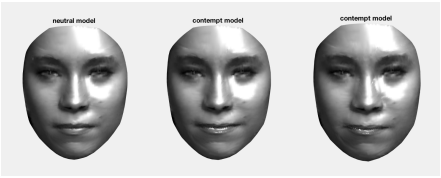


Figura 3: Applicazione dei due massimi di densità: la seconda faccia, quella creata utilizzando il secondo centroide, risulta essere meno credibile della prima

Questi test ci fanno capire quanto gli esempi del dataset siano concentrati in una piccola area nello spazio; questo fatto è probabilmente il principale motivo per cui le 3 strategie di media, mediana e moda creano le stesse facce, come è possibile vedere nel capitolo dei risultati.

3. Regressori

3.1. Regressore lineare

La prima tecnica, piú intuitiva, che è stata utilizzata per la predizione dei vettori rappresentanti le trasformazioni è la regressione lineare. Questa tecnica cerca di trovare il tipo di relazione funzionale (in questo caso lineare) che esiste tra variabili dipendenti e una o più variabili indipendenti per ottenere la funzione che descrive al meglio la distribuzione dei dati. Il modello di regressione lineare è:

$$Y_i = \beta_0 + \beta_1 X_i + u_i,$$

dove:

- i varia tra le osservazioni, $i = 1, \ldots, n$;
- Y_i è la variabile dipendente;
- X_i è la variabile indipendente o regressore;
- $\beta_0 + \beta_1 X$ è la retta di regressione o funzione di regressione della popolazione;
- β_0 è l'intercetta della retta di regressione della popolazione;
- β_1 è il coefficiente angolare della retta di regressione della popolazione;
- u_i è l'errore statistico.

Per la realizzazione è stata usata l'implementazione della libreria *scikit-learn*, che fornisce una classe python mediante la quale è possibile istanziare un regressore, specificando i dati su cui eseguire l'operazione di *fit* del modello ed ottenendo un risultato simile a quello dell'esempio in figura 4.

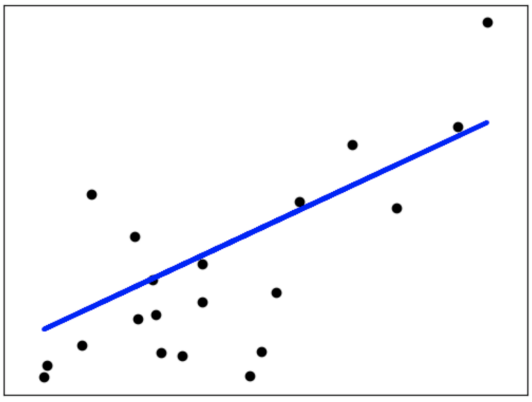


Figura 4: esempio di applicazione di regressore lineare

3.2. SVR (Support Vector Regression)

Le Support Vector Machines costituiscono una delle principali tecniche di predizione nell'ambito dell'apprendimento automatico, ed in particolare possono essere applicate non solo a problemi di classificazione ma anche ai casi in cui si trattano problemi risolvibili mediante regressione. Questo tipo di approccio mantiene le principali caratteristiche degli algoritmi di tipo SVM: tramite esempi di input e relative etichette di output composte da numeri reali, si cerca di massimizzare i margini di separazione con i vettori di supporto per creare delle funzioni non lineari che cercano di spiegare al meglio la distribuzione dei dati. Nel nostro caso, avevamo delle "multi etichette" - vettori di 300 elementi - e per questo è stato necessario predire usando un regressore *multioutput*, che ripete la stima per ogni elemento dell'array. La capacità del sistema è controllata da parametri che non dipendono dalla dimensionalità dello spazio delle features.

In figura 5 è possibile vedere un esempio di applicazione di SVR.

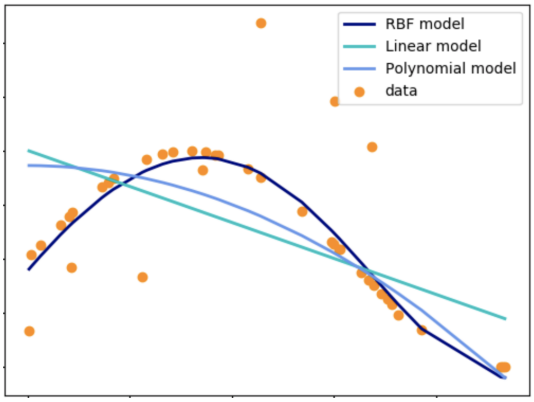


Figura 5: Esempio di applicazione di SVR

Come suggerito dall'immagine, l'efficienza del metodo dipende, in relazione alla disposizione spaziale dei dati presi in considerazione, dal kernel utilizzato per la predizione. Nel caso particolare è stata effettuata una 4-fold cross validation per determinare il miglior kernel da applicare per la risoluzione del problema ed i migliori valori per i parametri C ed ϵ per il regressore. Anche in questo caso è stata utilizzata l'implementazione fornita da *scikit-learn* per la realizzazzione della macchina.

3.3. Neural Network

Nel campo dell'apprendimento automatico, una rete neurale è un modello matematico composto da "neuroni" artificiali; queste interconnesioni vanno a formare la struttura della rete neurale, e definiscono una sorta di sistema adattivo che cambia i propri pesi associati in base a delle osservazioni costituite da un insieme di input e il relativo insieme di output. La fase di aggiornamento viene chiamata fase di training, e per cercare di essere più performante possibile la rete ha bisogno di un gran numero di campioni.

Le reti possono essere utilizzate per simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non riescono a rappresentare. Il generico modello strutturale di una Rete Neurale è schematizzato in figura 6.

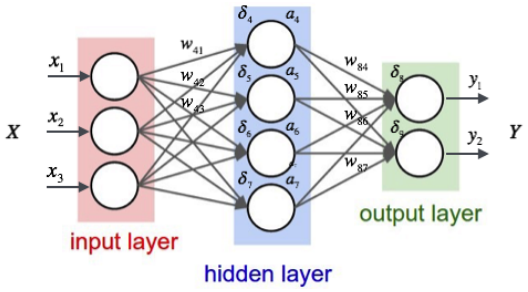


Figura 6: Modello di una Rete Neurale

Nel caso preso in esame è stata creata una rete di dimensioni ridotte, composta da 3 livelli, di cui solo uno nascosto (di 500 neuroni). Dato che si è trattato un problema di regressione, gli strati di imput e output hanno dimensione fissata, ognuno dei quali costituito da 300 nodi, pari alla lunghezza dei sample del dataset. Per l'implementazione è stata utilizzata la libreria di machine learning *Keras*[1] appoggiata da *Tensorflow*[3], e nello specifico la rete è composta da tre layer densi, due con activation function "relu" e una "tanh"; l'ottimizzatore scelto è "Adam".

3.4. Osservazioni

Per ogni tecnica di regressione è stato adottato un processo di cross validation per determinare sia la giusta proporzione tra insieme di train e insieme di test, sia per determinare i corretti iperparametri, ad esempio come già descritto nella sezione dell'SVR.

4. Risultati

4.1. Risultati Qualitativi

La procedura di valutazione delle tecniche studiate è stata quella di estrarre tutte le facce neutre dal dataset, prenderne alcune casuali e applicargli tutte le espressioni con tutte le tecniche secondo la formula:

$$deform_{vector} = neutral_{vector} + pred_{vector}(expr) * \alpha$$

dove α è un fattore moltiplicativo che cerca di accentuare l'espressione qualora questa sia di un tipo che modifica poco la faccia neutra (ad esempio: "contempt", "disgust", "angry" e "sadness"). Una volta calcolati i modelli espressivi con la relativa texture, li abbiamo visualizzati e salvati per valutare qualitativamente il risultato. Di seguito le immagini ottenute:

Angry

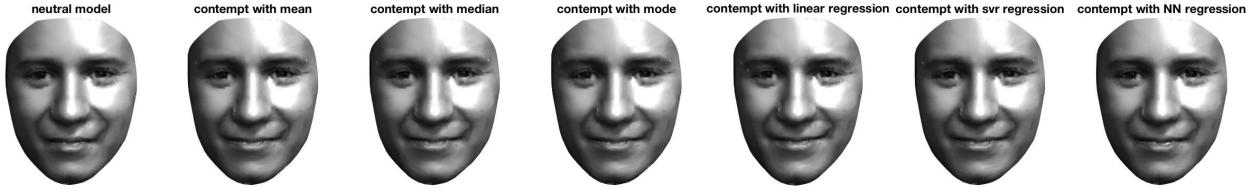
$\alpha = 1.0$

$\alpha = 1.5$

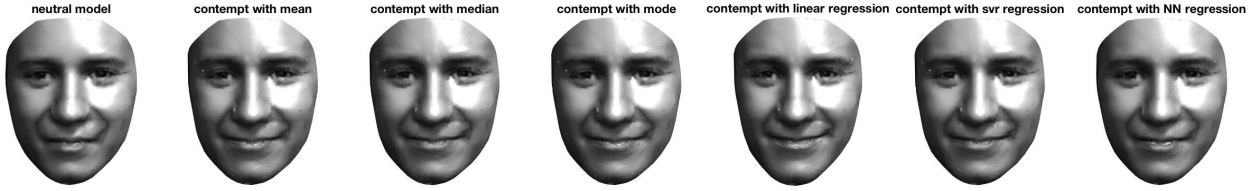
$\alpha = 2.0$

Contempt

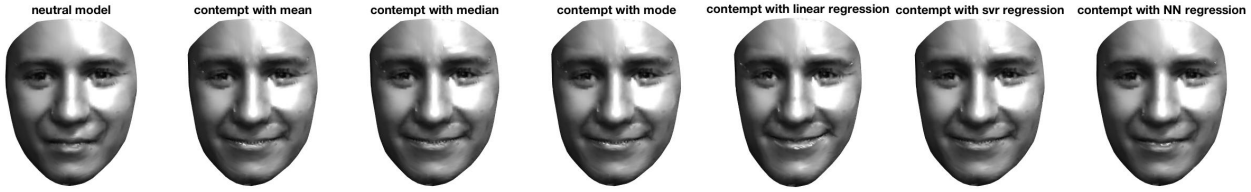
$\alpha = 1.0$



$\alpha = 1.5$

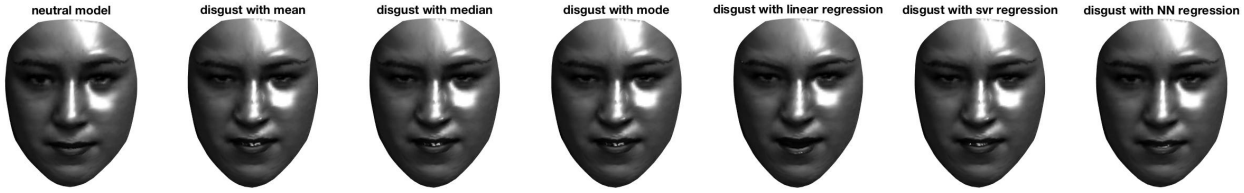


$\alpha = 2.0$

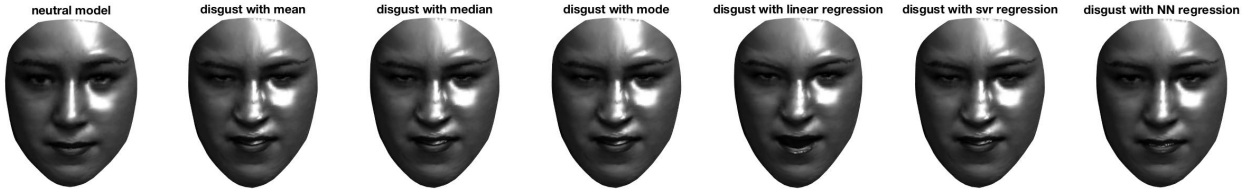


Disgust

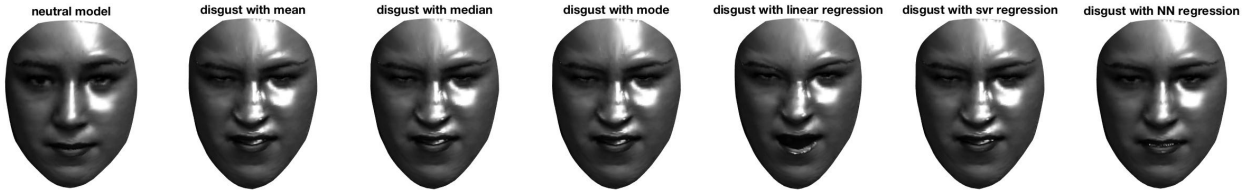
$\alpha = 1.0$



$\alpha = 1.5$



$\alpha = 2.0$



Fear

$\alpha = 1.0$

$\alpha = 1.5$

$\alpha = 2.0$

Happy

$\alpha = 1.0$

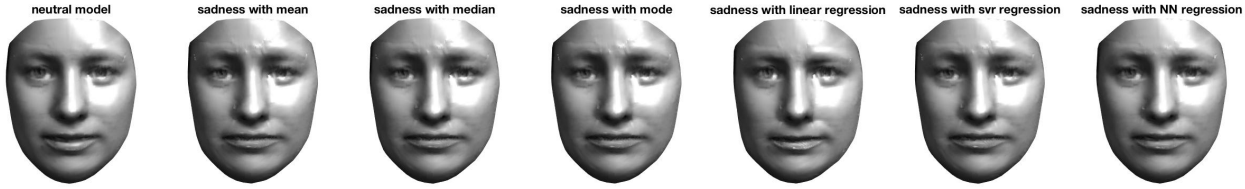
$\alpha = 1.5$

$\alpha = 2.0$

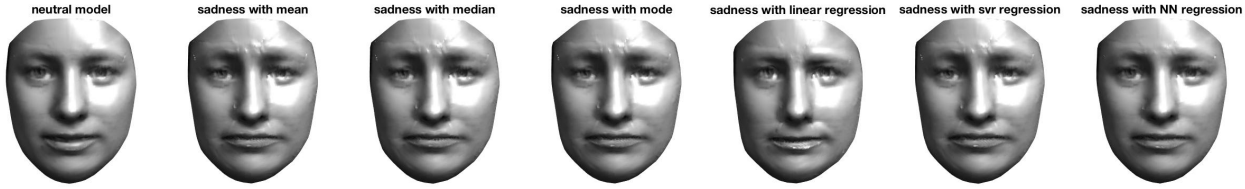
Sadness

$\alpha = 1.0$

$\alpha = 1.5$

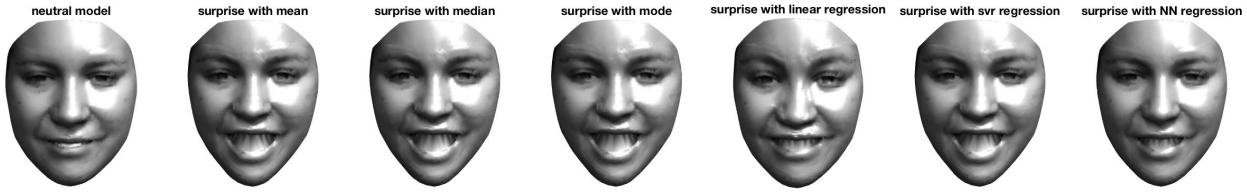


$\alpha = 2.0$

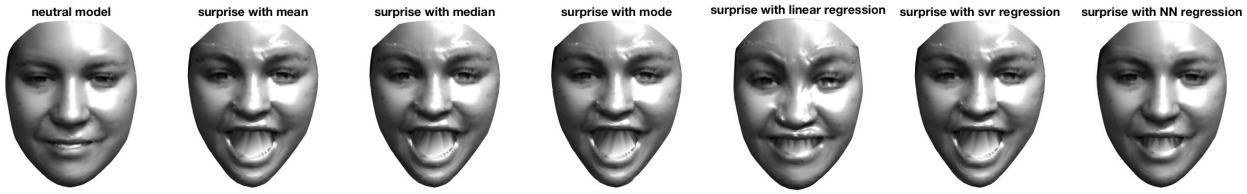


Surprise

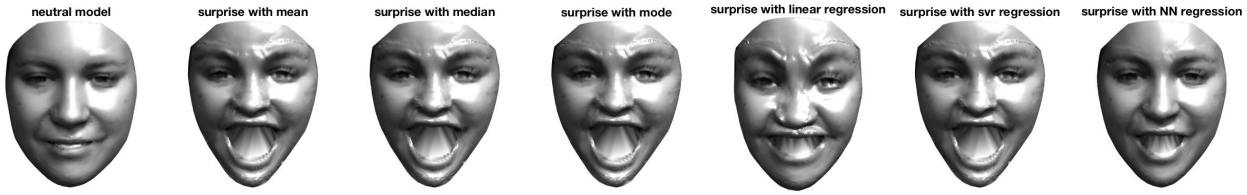
$\alpha = 1.0$



$\alpha = 1.5$



$\alpha = 2.0$



4.2. Risultati Quantitativi

Abbiamo effettuato anche delle analisi quantitative sull'errore commesso dai regressori per valutare più accuratamente quale fosse la tecnica più precisa. I risultati medi di questi test sono visibili nelle tabelle seguenti: la tabella 3 evidenzia gli errori quadratici medi che vengono commessi testando i regressori sull'insieme di test, calcolando quindi la differenza tra i vettori predetti e quelli reali; la tabella 4 mostra invece le varianze di questi errori.

	Regressore		
Espressione	Lineare	SVR	NN
happy	6.81	**3.61**	6.05
angry	5.52	**3.72**	4.77
disgust	6.16	**3.87**	5.41
fear	5.25	**3.92**	5.35
contempt	3.80	**2.62**	4.23
surprise	9.38	**4.44**	7.33
sadness	6.21	**4.22**	5.19

Tabella 3: La tabella mostra l'errore quadratico medio commesso sui test dalle varie tecniche

	Regressore		
Espressione	Lineare	SVR	NN
happy	2.98	0.75	**0.43**
angry	0.84	0.52	**0.38**
disgust	1.67	0.54	**0.43**
fear	0.54	0.30	**0.11**
contempt	0.68	**0.18**	0.31
surprise	11.84	**0.83**	1.52
sadness	2.48	**0.11**	0.18

Tabella 4: La tabella mostra la varianza dell'errore commesso sui test dalle varie tecniche

5. Conclusioni

Dai risultati quantitativi emersi dalle tabelle precedenti è evidente che, tra i regressori, la tecnica migliore sia l'SVR, in quanto è quella che commette meno errore medio con varianza relativamente piccola ma, data la particolare tipologia di problema, i risultati qualitativi sono quelli di maggior importanza. Dalle immagini appare abbastanza chiaro che le facce create dai metodi "statistici" siano molto simili data la disposizione dei dati (come già evidenziato nel capitolo dedicato); i regressori invece risultano essere

diversi tra di loro, con il lineare che molte volte deforma in facce innaturali, mentre la rete neurale non accentua abbastanza l'espressione (effetto sicuramente causato dalla scarsità dei dati per allenare il modello); per quanto riguarda invece l'SVR, si dimostra essere simile alla media, moda e mediana, con un risultato leggermente più naturale.

Lo studio del mean shift ha evidenziato che i vettori trasformazione sono tutti concentrati e molto vicini tra di loro, e che probabilmente i campioni sono tutti prossimi alla media (per questo le facce espresse dalla media, mediana e moda vengono somiglianti tra di loro). Questo fatto è confermato anche dalle espressioni ottenute tramite l'SVR: le facce create sono analoghe a quelle ottenute tramite le tecniche "statistiche"; dunque, non importa quale sarà l'espressione di partenza, la trasformazione sarà più o meno sempre la stessa e il regressore sarà simile a una funzione costante. Le espressioni ottenute con le varie tecniche, partendo da un dataset come il nostro, sono comunque abbastanza credibili, eventualmente da migliorare e accentuare attraverso un'$\alpha > 1$, sia usando semplicemente la moda (che comunque dovrebbe essere più precisa della media e mediana), sia con l'SVR per un risultato leggermente più naturale.

References

[1] Francois Chollet. *Keras*. `https://github.com/fchollet/keras`. 2015.

[2] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed ¡today¿]. 2001–. URL: `http://www.scipy.org/`.

[3] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[4] *MATLAB version Version 8.1 Optimization Toolbox*. The MathWorks, Natick, MA, USA. 2018. URL: `https://it.mathworks.com/help/matlab/matlab-engine-for-python.html`.

[5] *MATLAB version 9.4.0.813654 (R2018a)*. The Mathworks, Inc. Natick, Massachusetts, 2018.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.