UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA

Ridge regression / Kernel Ridge regression

SMML project:
Omar Ghezzi (Matr. 984352)

Learning problem

- **Obiettivo:** utilizzare un algoritmo di apprendimento (Ridge regression/Kernel ridge regression) per inferire un predittore h che approssimi la relazione che intercorre tra la features description di un brano musicale presente sulla piattaforma Spotify e la sua popolarità (target).

Regressione :

$$\mathcal{X} \equiv \mathbb{R}^d \ , \ \mathcal{Y} \equiv \mathbb{R}$$

$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$

Consideriamo la classe dei predatori lineari (iperpiani in d dimensioni) :

$$\mathcal{H} = \{h : h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^d\}$$

Algoritmo di apprendimento:

$$\begin{aligned} \boldsymbol{w}_{S_m} &= \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \sum_{t=1}^{m} (\boldsymbol{w}^T \boldsymbol{x} - y_t)^2 \\ &= (S^T S)^{-1} S^T \boldsymbol{y} \end{aligned}$$

Instabile, quando $S^T S$ ha almeno un autovalore nullo.
Perturbando leggermente l'istanza S_m , è possibile che il predittore prodotto dall'algoritmo non sia lo stesso:

$$\mathbb{E}\left[\ell(h_S, \boldsymbol{Z}'_t) - \ell(h_{S^{(t)}}, \boldsymbol{Z}'_t)\right]$$ Grande

h is a good proxy for $P(\mathrm{Y}|\mathrm{X})$ $\Leftrightarrow$ $\ell_{\mathcal{D}}(h) = \mathbb{E}_{(\boldsymbol{X},Y)\sim\mathcal{D}}\left[\ell(Y, h(\boldsymbol{X}))\right]$ is small.

Ridge regression: $\boldsymbol{w}_{S_m} = (\alpha \mathbf{I} + S^T S)^{-1} S^T \boldsymbol{y}$

Hyperparameter space: np.linspace(0, 15000, 150)

Nested 5-fold CV to estimate the (expected) risk

Pre-processing (1)

- Dato lo Spotify tracks dataset, considero due formulazioni:

  1) c=1 : feature description solo con features continue.
  2) c=2 : features description con features continue + binarie + ordinali (qualitative) + categoriche.

- Preprocessing:
  - Elimino colonne ‘Unnamed:0’ a ‘track ID’. (c=1, c=2)
  - Verifica missing values per ‘key’. (c=2)
  - Convertire feature ‘explicit’ booleana in binaria. (c=2)
  - Divisione examples multi-artista (-> q nuovi esempi per lo stesso brano, ciascuno con uno dei q diversi artisti come unico valore per la categoria ‘artists’). (c=2)
  - Carattere escape in ‘bad$$’. (c=2)
  - Permutazione casuale dell’intero dataset, utilizzando random seed. (c=1, c=2)
  - Rimozione di esempi ridondanti e inconsistenti. (c=1, c=2)

(X, Y)	Continuous Features										
	$X_1, \ldots, X_{10}, Y$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	Y
Name	duration_ms	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	Popularity
$\mathcal{X}, \mathcal{Y}$	$\mathcal{X}_1 = [0, 10^7]$	[0, 1]	[0, 1]	[-50, 5]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 250]	[0, 100]
Units of measure	ms	/	/	dB	/	/	/	/	/	BPM	/

X	Binary, ordinal (qualitative) and categorical features							
	$X_1, \ldots, X_5$							
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Name	key	time_signature	track_genre	explicit	mode	track_name	album_name	artists
$\mathcal{X}$	$\{-1, 0, 1, 2, ..., 11\}$	$\{3, 4, 5, 6, 7\}$	String	$\{T, F\}$	$\{0, 1\}$	String	String	String
# unique values	13	5	114	2	2	73609	46590	31438
Type	ordinal (qualitative)	ordinal (qualitative)	categorical	binary	binary	categorical	categorical	categorical

Pre-processing (2): Scaling (c=1, c=2)

- ERM (regression) -> (inverse) scale invariant $Z = DS \qquad \hat{\boldsymbol{w}_{S_m}} = D^{-1}\boldsymbol{w}_{S_m}$
- RERM (ridge regression) -> not scale invariant, a meno che $D^T D = \mathbf{I}$

$$\begin{aligned}
\hat{\boldsymbol{w}_{S_m}} &= \left(Z^T Z\right)^{-1} Z^T \boldsymbol{y} \\
&= \left(D^T S^T S D\right)^{-1} D^T S^T \boldsymbol{y} \\
&= D^{-1}\left(S^T S\right)^{-1} (D^T)^{-1} D^T S^T \boldsymbol{y} \\
&= D^{-1}\left(S^T S\right)^{-1} S^T \boldsymbol{y}
\end{aligned}$$

$$\begin{aligned}
\hat{\boldsymbol{w}_{S_m}} &= \left(Z^T Z + \alpha\mathbf{I}\right)^{-1} Z^T \boldsymbol{y} && (15)\\
&= \left(D^T S^T S D + \alpha\mathbf{I}\right)^{-1} D^T S^T \boldsymbol{y} && (16)\\
&= \left(D^T S^T S D + \alpha D^T (D^T)^{-1} D^{-1} D\right)^{-1} D^T S^T \boldsymbol{y} && (17)\\
&= \left(D^T \left(S^T S + \alpha (D^T)^{-1} D^{-1}\right) D\right)^{-1} D^T S^T \boldsymbol{y} && (18)\\
&= D^{-1}\left(S^T S + \alpha (D^T)^{-1} D^{-1}\right)^{-1} (D^T)^{-1} D^T S^T \boldsymbol{y} && (19)\\
&= D^{-1}\left(S^T S + \alpha (D^T)^{-1} D^{-1}\right)^{-1} S^T \boldsymbol{y} && (20)
\end{aligned}$$

In realtà, la presenza dell'iperparametro implicitamente assorbe $(D^T)^{-1}D^{-1}$ (a scale diverse dei dati potrà corrispondere predittore riscalato proporzionalmente, ma probabilmente non stesso valore di alpha)

Standardization (training): rendere ciascuna feature a media nulla e a deviazione standard unitaria (nel training set):

$$\overline{\boldsymbol{y}}^{(-i)} = \frac{\boldsymbol{y}^{(-i)} - \mu_{\boldsymbol{y}}^{(-i)}}{\sigma_{\boldsymbol{y}}^{(-i)}} \qquad \overline{\boldsymbol{x}}_j^{(-i)} = \frac{\boldsymbol{x}_j^{(-i)} - \mu_j^{(-i)}}{\sigma_j^{(-i)}} \quad \forall j = 1, ..., d$$

Standardization (testing): sottrarre la media e dividere per la deviazione standard ciascuna feature nel test set, utilizzando medie e deviazioni std calcolate nel training set:

$$\overline{\boldsymbol{x}}_j^{(i)} = \frac{\boldsymbol{x}_j^{(i)} - \mu_j^{(-i)}}{\sigma_j^{(-i)}}, \quad \text{and} \quad \overline{\boldsymbol{y}}^{(i)} = \frac{\boldsymbol{y}^{(i)} - \mu_{\boldsymbol{y}}^{(-i)}}{\sigma_{\boldsymbol{y}}^{(-i)}} \forall j = 1, ..., d$$

Pre-processing (3): Encoding (c=2)

	Target leakage	Dipendenza valori unici - numero features introdotte	Problemi tempo-Spazio	Altro
one-hot encoding	NO	Sì (gestione: [0|0|0|…|0|0|])	Sì	NO
Base N encoding	NO	Sì (gestione: [0|0|0|…|0|0|])	Sì (ma migliore di OHE)	NO
Hash encoding	NO	Sì (gestione: [0|0|0|…|0|0|])	NO	Gestione collisioni+encoding irreversibile
Count encoding	NO	Sì (gestione: 0)	NO	Se tutte le categorie hanno conteggio simile, risulta inutile
Target encoding	Sì, se ci sono tanti valori unici	Sì (gestione: costante*10-percentile dell'insieme delle medie target associate ai dati)	NO	Passando dal training al testing le medie potrebbero variare (non è però possibile tenere conto dei dati di testing per l'enconding)

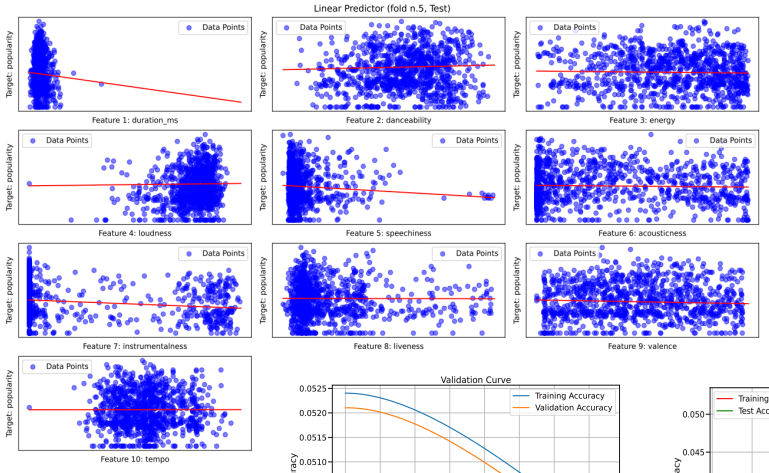
1. **target encoding** for 'track_genre', **count encoding** for 'artists', 'album_name', 'track_name'.
2. **one-hot encoding** for 'track_genre', **count encoding** for 'artists', 'album_name', 'track_name'.
3. **one-hot encoding** for 'track_genre', **hash encoding** for 'artists', 'album_name', 'track_name'.

Results (c=1)

- Ridge regression underfitta pesantemente: test e training accuracy sono entrambe basse ($\simeq$0), con regularization factor $\alpha = 0$

  - Prevedibile -> nessuna correlazione lineare tra alcuna feature e il target
- Scomponendo il predittore nelle sue d componenti -> $\simeq$ [0,0,0,…,0]

Folds	hyper. opt.	Training (acc.)	Test (acc.)	Avg. Training (acc.)	Avg. Test (acc.)
Fold n.0	0.0	0.058069	0.032509		
Fold n.1	0.0	0.052153	0.050667		
Fold n.2	0.0	0.052359	0.051353	0.052403	0.052104
Fold n.3	0.0	0.049899	0.062133		
Fold n.4	0.0	0.049535	0.063855		



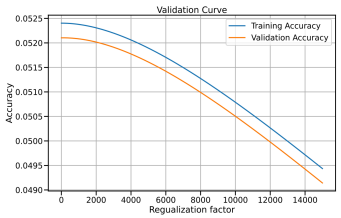
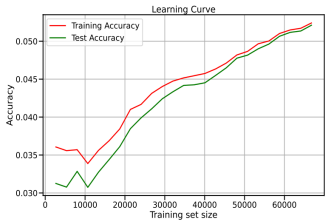
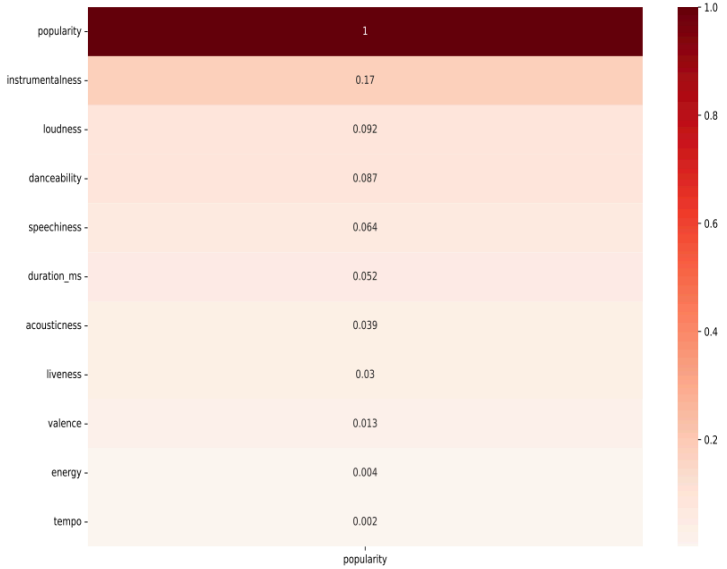


Figure 4: Validation curve ($c = 1$).



(a) Learning curve ($c = 1$)



Results (c=2)

- Variabili categoriche e ordinali (qualitative) diminuiscono il bias del modello (average accuracy≃0.5). Accuracy migliore con OHE per ‘track genre’ e count encoding per ’artists’, ’track_name’, ’album_name’. Regularization factor $\alpha = 100.67$

  - Il basso valore di accuracy per $\alpha = 0$ è probabilmente dovuto allo sparse encoding prodotto dalla OHE.

Folds	hyper. opt.	Training (acc.)	Test (acc.)	Avg. Training (acc.)	Avg. Test (acc.)
Fold n.0	0.000000	0.399457	0.393754		
Fold n.1	0.000000	0.396725	0.397675		
Fold n.2	0.000000	0.397175	0.396536	0.397405	0.395789
Fold n.3	0.000000	0.399001	0.394533		
Fold n.4	402.684564	0.394665	0.396447		

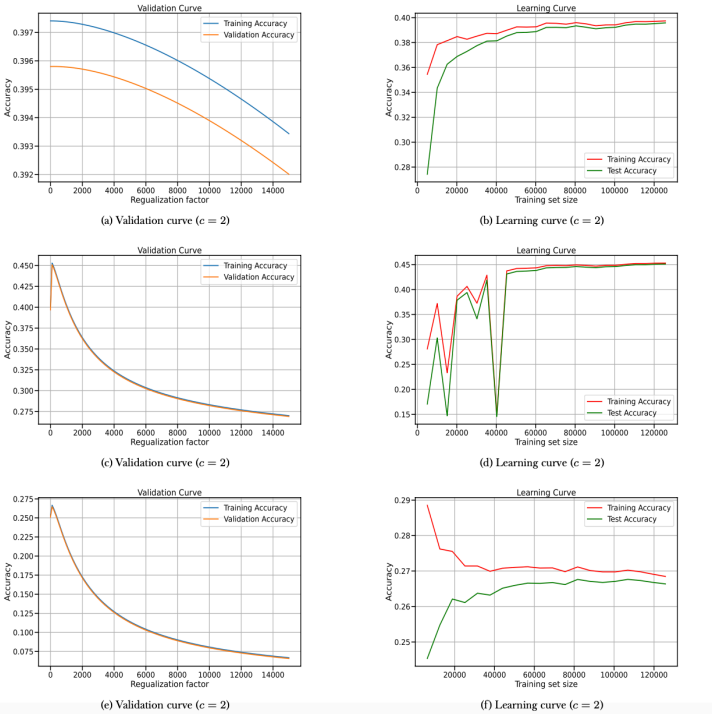
Table 5: 5-fold nested cross validation estimate with continuous, ordinal (qualitative), binary and categorical features (execution $c = 2$). Target encoding for 'track_genre' and count encoding for 'artists', 'track_name', 'album_name'.

Folds	hyper. opt.	Training (acc.)	Test (acc.)	Avg. Training (acc.)	Avg. Test (acc.)
Fold n.0	100.671141	0.453248	0.447921		
Fold n.1	100.671141	0.452226	0.452826		
Fold n.2	100.671141	0.453398	0.452717	0.452959	0.450925
Fold n.3	100.671141	0.455539	0.449489		
Fold n.4	100.671141	0.450384	0.451672		

Table 6: 5-fold nested cross validation estimate with continuous, ordinal (qualitative), binary and categorical features (execution $c = 2$). One-hot encoding for 'track_genre' and count encoding for 'artists', 'track_name', 'album_name'.

Folds	hyper. opt.	Training (acc.)	Test (acc.)	Avg. Training (acc.)	Avg. Test (acc.)
Fold n.0	100.671141	0.266936	0.264152		
Fold n.1	100.671141	0.266055	0.266884		
Fold n.2	100.671141	0.266589	0.264821	0.266686	0.26471
Fold n.3	100.671141	0.266143	0.266763		
Fold n.4	100.671141	0.267708	0.260930		

Table 7: 5-fold nested cross validation estimate with continuous, ordinal (qualitative), binary and categorical features (execution $c = 2$). One-hot encoding for 'track_genre' and hash encoding for 'artists', 'track_name', 'album_name'.



(a) Validation curve ($c = 2$) (b) Learning curve ($c = 2$)

(c) Validation curve ($c = 2$) (d) Learning curve ($c = 2$)

(e) Validation curve ($c = 2$) (f) Learning curve ($c = 2$)

Kernel ridge regression

- Ridge regression è un algoritmo parametrico, all'aumentare della taglia del campione soltanto l'estimation error diminuisce (nessuna possibilità di ridurre l'approximation error).
- È possibile diminuire approximation e estimation error rendendo un algoritmo non parametrico (che sceglie tra predittori descritti potenzialmente da un numero illimitato di parametri) consistente.

Feature expansion: trasforma datapoint in elementi di un RKHS, tale che un predittore h, in tale spazio, sia descritto da un numero (possibilmente infinito) di parametri.

$$\begin{aligned}\mathrm{Imm}(\Phi) &= \mathrm{Span}\big(\{\Phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}\big) \\ &= \Big\{f(\cdot) = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i) : n \in \mathbb{N}, \boldsymbol{x}_i \in \mathcal{X}, \alpha_i = \mathbb{R}\Big\} \\ &= \Big\{f(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, \boldsymbol{x}_i) : n \in \mathbb{N}, \boldsymbol{x}_i \in \mathcal{X}, \alpha_i = \mathbb{R}\Big\}\end{aligned}$$

$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

$$\forall \boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X}, \;\; (\boldsymbol{x}, \boldsymbol{x'}) \mapsto K(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{1}{2\gamma}||\boldsymbol{x}-\boldsymbol{x'}||^2}$$

$$\mathcal{H}_K = \mathrm{Imm}(\Phi) \bigcup \Big\{h(\cdot) : h(\cdot) = \lim_{n\to\infty}\{f_n(\cdot)\}, \;\; f_1(\cdot), f_2(\cdot), \ldots \in \mathrm{Imm}(\Phi)\Big\}$$

Apprendere un predittore non-lineare nello spazio originario (ma lineare nel feature space espanso) permette ad un algoritmo (parametrico in principio, ora reso non parametrico) di raggiungere la consistenza.

Kernel ridge regression :

$$f^*(\boldsymbol{x}) = \langle \sum_{t=1}^{m} \alpha_t^* K(\boldsymbol{x}_t, \cdot), K(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}_K} = \boldsymbol{y}^T (\lambda \mathbf{I}_{m\times m} + \boldsymbol{K})^{-1} \boldsymbol{K}_{\boldsymbol{x}'}$$

$$\boldsymbol{\alpha}_{S_m}^* = \boldsymbol{y}^T (\lambda \mathbf{I} + \boldsymbol{K})^{-1} \quad , \quad \text{so that:} \quad f^*(\boldsymbol{x}) = \boldsymbol{\alpha}_{S_m}^{*T} \boldsymbol{K}_{\boldsymbol{x}'}$$

Subsampling - DCKRR

Calcolare $(\lambda \mathbf{I}_{m\times m} + \boldsymbol{K})^{-1}$ richiede complessità temporale $\mathcal{O}(m^3)$ e spaziale $\mathcal{O}(m^2)$, perciò:

1) esecuzione dell'algoritmo su $S_{r'}$ con $r' = 4000$ esempi.
2) esecuzione, sull'intero dataset S_r della versione DCKRR:

   - Ogni training part $S_m^{(-i)}$ è divisa in $M = 20$ sottoparti, da $m' \simeq 3200$ esempi ciascuno, ed è usata per ottenere altrettante predizioni (se si considera fissato l'i-esimo fold e il datapoint di test corrente):

$$f_\theta^{(-i,j)}(x') = y^T(\theta I + K^{(-i,j)})^{-1}K_{x'}^{(-i,j)} \quad x' \in S_n^i$$

   - la predizione media prende parte al calcolo della loss per il calcolo del test error per l'i-esimo fold:

$$\bar{f}_\theta^{(-i)}(x') = \frac{1}{M}\sum_{j=1}^{M} f_\theta^{(-i,j)}(x') \qquad \ell_{S_n^{(i)}}(\bar{h}_\theta^{(i)}) = \frac{1}{n}\sum_{(x',y')\in S_n^{(i)}}^{M} \ell(y, \bar{f}_\theta^{(-i)}(x'))$$

Entrambi i metodi si servono della PLU-factorization per calcolare l'inversa di $(\lambda \mathbf{I}_{m\times m} + \boldsymbol{K})^{-1}$

```
P, L, U = sp.linalg.lu(self.alpha*ID + K)

self.w = np.dot(y.T, sp.linalg.solve(U, sp.linalg.solve(L, P))).reshape(-1)
```

Results

c=1

Gamma	Alpha	Max. training acc. estimate	Max. test acc. estimate
0.00001	1000.0	0.999996	-0.002625
0.00100	1000.0	0.999117	-0.002629
0.01000	1000.0	0.998590	-0.002629
0.10000	1000.0	0.997688	-0.002626
1.00000	10.0	0.774916	0.032003
2.00000	10.0	0.545614	0.044065
4.00000	5.0	0.340123	0.050967
5.00000	5.0	0.290289	0.052553
10.00000	5.0	0.182262	0.054414
15.00000	1.0	0.144970	0.054841
20.00000	1.0	0.126388	0.055644
50.00000	0.5	0.090542	0.054608
100.00000	0.1	0.076321	0.054833
500.00000	0.1	0.045663	0.036767
4000.00000	0.1	0.033401	0.027326
20.0	1.0	0.089439	0.055644

c=2 ("target encoding" for "track_genre" and "count encoding" for "artists", "album_name" and "track_name")

Gamma	Alpha	Max. training acc. estimate	Max. test acc. estimate
0.00001	1.0	0.998719	-0.000240
0.00100	1.0	0.998411	0.000774
0.01000	0.5	0.998353	0.002127
0.10000	0.1	0.998346	0.009407
1.00000	0.1	0.998306	0.079082
2.00000	0.1	0.947981	0.168577
4.00000	1.0	0.841538	0.232828
5.00000	5.0	0.795044	0.238412
10.00000	5.0	0.646683	0.278079
15.00000	5.0	0.573672	0.285697
20.00000	5.0	0.530968	0.284819
50.00000	1.0	0.442073	0.275469
100.00000	0.5	0.410365	0.251253
500.00000	0.1	0.360595	0.219017
4000.00000	0.1	0.338284	0.215640
15.00000	5.0	0.573672	0.285697

```
alphas = np.array([0.001, 0.1, 0.5, 1,  5, 10, 50, 100, 1000])
gammas = np.array([0.00001, 0.001, 0.01, 0.1, 1, 2, 4, 5, 10, 15, 20, 50, 100, 500, 4000])
hyper_space = np.array(list(product(alphas, gammas)))
```

M	m'	Gamma	Alpha	Max. training acc. estimate	Max. test acc. estimate
20	3200	10	5	0.110672	0.110108
20	3200	20	1	0.115148	0.114499
20	3200	20	1	0.115148	0.114499

Table 2: Continuous features ($c = 1$). 5-fold best CV estimate with A =DCKRR and $\Theta_0 = [(\alpha_0, \gamma_0), (\alpha_1, \gamma_1)] = [(5, 10), (1, 20)]$.

M	m'	Gamma	Alpha	Max. training acc. estimate	Max. test acc. estimate
36	3200	15	5	0.479499	0.477584
18	7000	15	5	0.49681	0.494555
11	10000	15	5	0.505889	0.503348

Table 4: Continuous, binary, ordinal and categorical features ($c = 2$). 5-fold best CV estimate with A =DCKRR (several sub-sizes m') and $\Theta_0 = [(\alpha_0, \gamma_0)] = [(5, 15)]$.