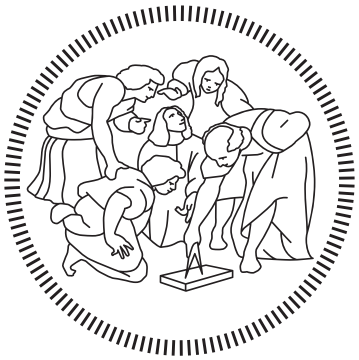
POLITECNICO DI MILANO
Facoltà di Ingegneria
Scuola di Ingegneria dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria
Corso di Laurea Magistrale in Ingegneria Informatica



# INVERSE REINFORCEMENT LEARNING E CATEGORIZZAZIONE DI UTENTI

## Analisi di consumi idrici in ambiente domestico

Relatore: Prof. Marcello Restelli
Correlatori: Dott. Ing. Andrea Cominola
Dott. Matteo Pirotta

Tesi di Laurea Magistrale di:
Matteo Gallesio, matricola 800670

Anno Accademico 2015-2016

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Abstract

L'obiettivo del problema di *Inverse Reinforcement Learning* è individuare, a partire dal comportamento di un agente, una rappresentazione sotto forma di *funzione premio* degli obiettivi da esso perseguiti. In un contesto multiagente, è possibile confrontare le funzioni premio dei vari agenti e suddividerli, ad esempio mediante tecniche di *clustering*, in categorie, ognuna delle quali è costituita da agenti che condividono funzioni premio simili.

Lo scopo di questa tesi è proporre un approccio che combina tecniche IRL – nello specifico, l'algoritmo *Gradient Inverse Reinforcement Learning* (*GIRL*) – con metodi di clustering per categorizzare gli utenti di un servizio sulla base dei loro interessi ed obiettivi. Le due fasi sono separate: prima, mediante GIRL, individuiamo le funzioni premio degli agenti; poi, mediante un algoritmo di clustering, individuiamo le categorie di utenti.

Il nostro caso di studio specifico, che considera dati reali, è la modellizzazione e la categorizzazione di utenti residenziali sulla base delle caratteristiche dei loro consumi idrici. Impieghiamo una funzione premio che punta a modellizzare le abitudini degli utenti, e compariamo i risultati da essa ottenuti con quelli di uno studio precedente simile al nostro ed effettuato sui nostri stessi dati.

Dimostriamo che le categorie di utenti individuate dal nostro procedimento sono caratterizzate da abitudini di consumo ben distinguibili l'una dall'altra e descrivibili per mezzo delle proprie funzioni premio.

---

The goal of the *Inverse Reinforcement Learning* problem is to obtain, on the basis of an agent's behavior, a *reward function* which represents its objectives. In a multi-agent context, it is possible to compare the reward functions of different agents and to separate them, i.e. with *clustering* techniques, into categories, each consisting of agents which share a similar reward function.

The goal of this thesis is to propose an approach which combines IRL techniques – specifically, the *Gradient Inverse Reinforcement Learning* (*GIRL*) algorithm – with clustering methods in order to categorize the users of a service on the basis of their interests and objectives. The two phases are separate: first, using GIRL, we find out the agents' reward functions; then, using a clustering algorithm, we find out the categories of users.

Our specific case study, which considers real data, is the modelization and categorization of residential users on the basis of the characteristics of their water consumption. We use a reward function which aims to model the users' habits, and we compare its results with those of a previous similar study based on the same data as ours.

We show that the categories found by our procedure are characterized by consumption habits well distinguished from one another, and can be described using their reward functions.

# Summary


*RL and IRL*

The basic assumption of *Reinforcement Learning* (RL) (Sutton and Barto, 1998) is that an *agent* operating in a specific environment (such as the user of a service) acts on the basis of specific preferences, which can be described by a *reward function* $R(s, a)$ which associates to a certain *action* performed in a certain *state* a *reward*, an abstract representation of something which the agent considers "good".

The goal of *Inverse Reinforcement Learning* (IRL) (Russell, 1998) is to obtain, starting from the agent's observed behavior, such reward function, which is not always intuitive to define manually. This approach can lead to a greater understanding of the agent when compared to one whose goal is simply to obtain a *policy* which merely imitates the agent's actions: similar objectives can lead to different behaviors, and different objectives can lead to similar behaviors. The reward function is often considered to be a better, more concise description of an agent's behavior than its policy, and can be used to generalize such behavior to areas of the state space not explored by the agent itself. However, the basic IRL problem is ill-posed due to its *ambiguity*, since several different reward functions can be used to explain the very same behavior. Different algorithms deal with this problem in different ways.

A common assumption (Ng and Russell, 2000) is to use a linearly parametrized reward function – $R(s, a) = \boldsymbol{\omega} \cdot \boldsymbol{\phi}(s, a)$ – which can be interpreted as the weighted sum of a set of *basis functions* or *features*: each feature represents one of the agent's objectives, and its weight represents how important it is considered by the agent. Starting from features it is possible to define *feature expectation* (Abbeel and Ng, 2004), that is, the expected value of the reward accumulated with respect to a certain feature given a policy.

Several approaches have been proposed in literature to solve the IRL problem. Some aim to minimize a metric based on the difference between the feature expectation of the agent's policy and the one of the proposed policy; others treat the IRL problem as a classification problem whose aim is to "label" states with an appropriate action; others aim to maximize the likelihood of the agent's behavior; and another group employs probabilistic methods to infer the agent's reward function. In particular, the *Gradient Inverse Reinforcement Learning* (*GIRL*) (Pirotta and Restelli, 2016) – which is used in this thesis – aims to find a reward function which minimizes the norm of the gradient of the policy performance, thus making the agent optimal. In order to apply GIRL, the policy must be stochastic, parametric and differentiable.

*Extension to multiple users*

The intuitive meaning of the reward function makes it particularly useful in multi-agent contexts: we can suppose that agents with similar objectives can be described by similar reward functions. It is thus possible to *cluster* agents on the basis of their reward functions. A possible application of this is *customer segmentation*: for example, an e-commerce site could keep track of the products viewed and purchased by users in order to present them with personalized advertisement or offers.

*Objectives of this thesis*

The goal of this thesis is to propose an approach which combines IRL and clustering in order to discover categories of agents with similar behaviors. Some similar approaches already exist (Babeş-Vroman et al., 2011; Choi and Kim, 2012), but their aim is to cluster single trajectories – that is, single state/action sequences – as opposed to full agents. We propose a simple procedure based on the GIRL algorithm: we first gather and analyze user data, obtaining information which can then be used to design an appropriate policy and reward function; we then run the GIRL algorithm to obtain the weights or said reward function; finally, we apply a clustering algorithm to the found weights and check if the clusters are indeed representative of actual user groups.

Our specific case study is the modelization of home users on the basis of their water consumption (Cominola et al., 2015), with the aim of discovering their *drivers* – that is, the features and motivations of their consumption. This kind of information can be used by water utilities to, for example, discover users prone to wasting water, or even inefficiencies and faults in the distribution network, and then act accordingly by applying appropriate *water demand management strategies*.

Our data consists of 29 weeks of hourly consumptions, labeled with hour of the day and day of the week, from 175 households located in Tegna, district of Locarno, Switzerland.

To represent users, we chose a combined Bernoulli/lognormal stochastic policy: for each data point, the Bernoulli portion is used to determine whether a user consumes water, and if this is the case the lognormal part is used to generate a consumption value. We used time, day and several past consumptions as features. We have shown that this kind of policy is able to represent the actual users' consumption data with a fair degree of accuracy.

For the reward function, we tried to come up with features able to represent the agents' routines, thus following a direction similar to the eigenbehavior-based one of (Cominola et al., 2016). We used a linearly parametrized reward functions whose features relate the current consumption to past ones; in addition to these, the simple negation of the current consumption acts as a cost function to minimize. We then used an Expectation-Maximization method based on a Gaussian Mixture Model to cluster agents on the basis of their reward functions.

Comparing our results with the ones of (Cominola et al., 2016), we have confirmed that our clustering based on reward functions is indeed able to distinguish users into categories, each of which – albeit with a limited number of outliers – is characterized by a recognizable shared routine such as a marked preference for medium or null consumptions. One of these categories, in particular, contains most households with little to no actual consumptions. We have thus proved the usefulness of our approach.

As our approach is rather simple, there is definitely room for improvement: for example, other kinds of categorizations could be tried, more complex models could be used to describe the data, and a more refined algorithm which performs clustering directly during the IRL phase could be designed, akin to some others already proposed in literature as noted above.

# Riconoscimenti

Questa tesi è stata svolta in ambito collaborativo con il gruppo di Natural Resources Management del professor Andrea Castelletti al Dipartimento di Elettronica, Informazione e Bioingegneria.

This work was partially supported by the *SmartH2O: an ICT Platform to leverage on Social Computing for the efficient management of Water Consumption* research project funded by the EU Seventh Framework Programme under grant agreement no. 619172.

# 1 Introduzione

*RL e IRL*

L'assunzione alla base del campo dell'*apprendimento per rinforzo* (*Reinforcement Learning*, RL) (Sutton e Barto, 1998) è che un agente che opera in un determinato ambiente (ad esempio, l'utente di un servizio) agisce sulla base di specifiche preferenze, descrivibili con una *funzione premio* (*reward function*) $R(s, a)$ che associa ad una certa azione compiuta in date condizioni un *premio*.

L'obiettivo dell'*apprendimento per rinforzo inverso* (*Inverse Reinforcement Learning*, IRL) Russell, 1998 è individuare a partire dal comportamento dell'agente tale funzione premio, la cui definizione a priori non è sempre intuitiva. Questo approccio può portare ad una comprensione maggiore dell'agente rispetto ad uno che punta ad ottenere una *politica* che si limita ad imitare le azioni compiute: comportamenti diversi possono essere infatti motivati dagli stessi obiettivi, e comportamenti simili possono essere motivati da obiettivi differenti. Inoltre, la funzione premio stessa può in alcuni contesti essere l'obiettivo della ricerca.

Un'assunzione comune nel campo di IRL (Ng e Russell, 2000) è che la funzione premio sia descrivibile come combinazione – anche lineare – di diverse *funzioni base* o *feature*, ognuna delle quali rappresenta uno degli obiettivi che l'agente punta a perseguire: individuare i parametri della funzione premio significa quindi scoprire quali obiettivi sono effettivamente perseguiti dall'agente e quanta importanza relativa è assegnata a ciascuno di essi.

*Contesto multiagente*

Questa caratteristica della funzione premio è particolarmente utile in un contesto multiagente: si può supporre che agenti che perseguono obiettivi simili siano descrivibili con funzioni premio simili. È quindi possibile effettuare una *categorizzazione* (*clustering*) degli agenti sulla base delle loro funzioni premio. Ciò permette ad esempio di effettuare *customer segmentation*: un sito di e-commerce può tener traccia dei prodotti visualizzati e acquistati dagli utenti per proporre loro pubblicità o offerte mirate.

*Obiettivi di questa tesi*

Lo scopo di questa tesi è proporre un approccio che combina metodi di IRL e clustering per categorizzare gli utenti di un servizio sulla base dei loro interessi ed obiettivi. In letteratura esistono algoritmi basati su questa stessa idea (Babeş-Vroman et al., 2011; Choi e Kim, 2012), ma a differenza del nostro approccio puntano solitamente a categorizzare singole *traiettorie* – cioè, singole sequenze stato/azione – senza quindi considerare gli agenti come entità atomiche. A tal fine, utilizziamo l'algoritmo *Gradient Inverse Reinforcement Learning* (*GIRL*) (Pirotta e Restelli, 2016), che opera cercando di individuare una funzione premio tale da rendere *ottima* la politica dell'agente. Il nostro procedimento consiste semplicemente nell'applicare tecniche di IRL per individuare prima la politica e poi la funzione premio di ogni agente, e successivamente applicare un algoritmo di clustering a tali funzioni premio al fine di individuare agenti con comportamenti simili.

Il nostro caso di studio specifico, che considera dati reali, è la modellizzazione e la categorizzazione di utenti residenziali sulla base delle caratteristiche dei loro consumi idrici (Cominola et al., 2015), con l'obiettivo di individuare le motivazioni alla base di tali consumi. Questo tipo di informazioni può essere sfruttato dalle compagnie di

distribuzione per individuare, ad esempio, utenti disattenti o propensi allo spreco, o anche perdite o guasti nella rete, e di conseguenza prendere misure applicando strategie di gestione della domanda idrica (*water demand management strategies*, WDMS) appropriate.

Come termine di paragone, abbiamo sfruttato uno studio (Cominola et al., 2016) che utilizza gli stessi nostri dati e punta ad individuare regolarità nel comportamento degli utenti analizzando i loro *eigenbehavior* (Eagle e Pentland, 2009), cioè le componenti principali di tale comportamento. Abbiamo effettuato un esperimento con una funzione premio che punta a rappresentare tale regolarità correlando il consumo corrente a determinati consumi passati: il risultato del clustering finale mostra che le categorie sono effettivamente composte da utenti con abitudini simili, e ognuna di esse è caratterizzata da un comportamento specifico.

Pur portando a risultati validi, il nostro approccio è molto semplificato, sia come struttura che come scelte progettuali ed implementative. Potrebbe essere arricchito, ad esempio, progettando un algoritmo che come quelli sopra citati integra le fasi di IRL e clustering in modo più stretto, oppure utilizzando classi di politica e feature più complesse, o ancora cercando di individuare altre caratteristiche degli utenti non necessariamente collegate alla regolarità del loro comportamento.

*Struttura della tesi*

La tesi è strutturata come segue:

Nel capitolo 2 sono descritti i fondamenti del problema di IRL, è presentata una panoramica sintetica degli algoritmi presenti in letteratura ed è introdotto il concetto di clustering insieme ad alcuni algoritmi IRL sviluppati per un contesto multiagente.

Nel capitolo 3 è descritto il nostro contesto applicativo specifico, cioè il processo di modellizzazione degli utenti di una rete idrica. È inoltre presentato come termine di paragone uno studio parallelo al nostro.

Nel capitolo 4 è delineato il processo generale seguito durante questa tesi.

Nel capitolo 5 sono descritte e motivate le nostre scelte di modello e sono presentati e valutati i risultati ottenuti.

Nel capitolo 6 sono riassunti sinteticamente i nostri risultati, e sono presentati possibili sviluppi futuri di questa linea di ricerca.

# 2 Stato dell'arte: Inverse Reinforcement Learning

In questo capitolo saranno introdotti i concetti alla base del problema di Inverse Reinforcement Learning e alcuni approcci rilevanti impiegabili nella sua soluzione. Particolare attenzione sarà rivolta all'algoritmo *GIRL* e ad alcuni metodi applicabili al caso di agenti multipli.

## 2.1 Processi decisionali di Markov (MDP)

Prima di poter descrivere nel dettaglio il contesto di Inverse Reinforcement Learning, è necessario introdurre il concetto basilare di *processo decisionale di Markov* (Markov Decision Process, MDP): si tratta di un formalismo matematico utilizzato per modellizzare problemi di decisione sequenziale, fornendo una rappresentazione dell'ambiente (concreto o astratto) in cui un *agente* (ad esempio, l'utente umano di un sistema o un processo software autonomo) opera e si sposta.

### 2.1.1 Definizione

Un MDP è definibile (Sutton e Barto, 1998) come la combinazione di:

- un insieme $S$ di *stati* in cui l'agente può trovarsi. Su tale insieme può essere definita una *distribuzione di stati iniziali* $D : S \to [0, 1]$, che rappresenta la probabilità di iniziare l'esplorazione in un determinato stato.
- un insieme $A$ di *azioni* che l'agente può eseguire.
- una *funzione di transizione* (*transition function*) $T : S \times A \times S \to [0, 1]$, che dati uno stato $s$, un'azione $a$ e uno stato $s'$ rappresenta la probabilità che l'azione $a$ eseguita nello stato $s$ porti l'agente nello stato $s'$, cioè $T(s, a, s') = P(s'|s, a)$.
- una *funzione premio* (*reward function*) $R : S \times A \to \mathbb{R}$ che rappresenta il *premio*, cioè una rappresentazione di quanto un agente "guadagna" (o perde, nel caso di premio negativo) eseguendo una determinata azione in un determinato stato. La funzione premio può anche essere definita come dipendente unicamente dallo stato ($R : S \to \mathbb{R}$), a seconda del dominio applicativo.
- un *fattore di sconto* (*discount factor*) $\gamma \in (0, 1)$, che funge da fattore di riduzione per i premi lontani, rappresentando una preferenza più o meno marcata da parte dell'agente per premi vicini nel tempo.

Un esempio molto semplice di MDP è descritto in Figura 2.1.

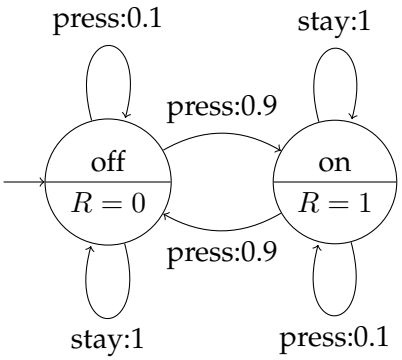



Figura 2.1: Semplice esempio di MDP, rappresentante un interruttore difettoso. I due stati sono *off* (spento) e *on* (acceso), l'unico stato iniziale è *off* ($D(\textit{off}) = 1$). Sono a disposizione le due azioni *stay* (non fare nulla), che ha sempre successo ($T(s, \textit{stay}, s) = 1$), e *press* (premi l'interruttore), che ha il 10% di probabilità di fallire ($T(\textit{on}, \textit{press}, \textit{off}) = T(\textit{off}, \textit{press}, \textit{on}) = 0.9$). In questo caso i premi sono associati agli stati, ed è modellizzato il fatto che l'obiettivo è accendere l'interruttore.

*Proprietà di Markov*

Una struttura di questo tipo è caratterizzata dalla *proprietà di Markov*: le funzioni transizione e premio valutate in un determinato stato non dipendono dalla *traiettoria* (cioè dalla sequenza {stato $\rightarrow$ azione $\rightarrow$ premio $\rightarrow$ stato $\rightarrow \dots$}) percorsa dall'agente per raggiungerlo, ma unicamente dallo stato stesso. Formalmente:

$$P(s_{n+1}, r_n | \{s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_n\}, a_n) = P(s_{n+1}, r_n | s_n, a_n)$$

### 2.1.2 Valutazione del comportamento di un agente

Il comportamento di un agente in un ambiente è descrivibile in termini di *politica* (policy), una funzione $\pi : S \rightarrow A$ che associa ad ogni stato l'azione scelta in esso dall'agente. La politica è anche descrivibile in termini stocastici come $\pi : S \times A \rightarrow [0, 1]$: in questo caso, essa definisce una distribuzione di probabilità sullo spazio delle azioni per ogni stato del sistema.

L'agente comincia la sua esplorazione in uno stato estratto dalla distribuzione di stati iniziali, e agendo secondo la politica scelta percorrerà una certa traiettoria e accumulerà una certa quantità di premio.

*Funzioni valore*

Fissati un MDP e una politica, è possibile definire le sue *funzioni valore* (Sutton e Barto, 1998):

- la *funzione valore dello stato* (funzione V) rappresenta l'*utilità* di uno stato, valutata sulle traiettorie ottenute a partire da esso e seguendo una politica $\pi$:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i R(s_i, a_i) | s_t = s\right] \tag{2.1}$$

- la *funzione valore dell'azione* (funzione Q) rappresenta l'utilità di eseguire una determinata azione in un determinato stato, ancora valutata sulle traiettorie indotte a partire da esso dalla politica $\pi$:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i R(s_i, a_i) | s_t = s, a_t = a\right] \tag{2.2}$$

Tali funzioni possono essere scritte analiticamente sotto forma di *equazioni di Bellman*: *Equazioni di Bellman*

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi}\left[R(s,a) + \gamma\mathbb{E}_{s'\sim T(s,a,\cdot)}V^{\pi}(s')\right] \tag{2.3}$$

$$Q^{\pi}(s,a) = R(s,a) + \gamma\mathbb{E}_{s'\sim T(s,a,\cdot)}V^{\pi}(s') \tag{2.4}$$

Una politica si dice *ottima* se, seguendola, l'agente massimizza il premio accumulato lungo le traiettorie da essa indotte, cioè se sono verificate le *equazioni di ottimalità di Bellman* derivate dalle funzioni valore: *Politica ottima*

$$V^{*}(s) = \max_{a}\left[R(s,a) + \gamma\mathbb{E}_{s'\sim T(s,a,\cdot)}V^{*}(s')\right] \tag{2.5}$$

$$Q^{*}(s,a) = R(s,a) + \gamma\mathbb{E}_{s'\sim T(s,a,\cdot)}\max_{a'}Q^{*}(s',a') \tag{2.6}$$

## 2.2 Risoluzione di un MDP: Reinforcement Learning

Risolvere un MDP significa individuare una sua politica ottima, ad esempio individuando la sua funzione Q ottima $Q^{*}$.

Se il modello dell'ambiente – cioè, le sue funzioni $T(s,a,s')$ e $R(s,a)$ – è completamente noto è possibile impiegare la *programmazione dinamica* (*Dynamic Programming*, *DP*) (Bellman, 1957), che tuttavia in caso di MDP di grandi dimensioni risulta computazionalmente onerosa. Non essendo impiegato in questa tesi, tale metodo non sarà descritto approfonditamente.

Se, al contrario, il modello dell'ambiente non è noto, è possibile impiegare metodi di *apprendimento per rinforzo* (Reinforcement Learning, RL).[1] Tali algoritmi possono essere categorizzati secondo diverse tassonomie a seconda delle loro proprietà: *Tassonomie basate sulle proprietà*

- i metodi *model-based* operano su un'approssimazione dell'ambiente costruita a partire dai dati disponibili, mentre i metodi *model-free* non costruiscono una rappresentazione esplicita dell'ambiente e operano direttamente sui dati.

- i metodi *on-policy* valutano la stessa politica usata per compiere le decisioni, mentre i metodi *off-policy* usano per compiere decisioni una politica distinta da quella in corso di valutazione.

- i metodi *online* operano direttamente sui dati dell'ambiente e possono interagire con esso o con una sua versione simulata, mentre i metodi *offline* o *batch* hanno a disposizione una quantità limitata di dati raccolti da una politica esplorativa e per un corretto funzionamento richiedono che tale esplorazione sia sufficientemente esaustiva.

Queste categorie possono essere combinate tra loro in vari modi, con l'eccezione della combinazione offline/on-policy per definizione impossibile. In questa tesi abbiamo a disposizione solo dati storici, senza possibilità di adoperare simulatori: pertanto, utilizzeremo solo metodi offline.

[1] Per una trattazione più approfondita si rimanda a (Sutton e Barto, 1998) e (Busoniu et al., 2010).

*Tassonomie basate sulle caratteristiche tecniche*

È possibile categorizzare gli algoritmi RL sulla base delle loro caratteristiche tecniche. In generale, tutti i metodi possono essere ricondotti ad uno schema *actor-critic*, che consiste in un "attore", cioè un componente con il compito di prendere decisioni aggiornando la politica o selezionando una data azione, e in un "critico", che fornisce all'attore le informazioni necessarie.

Gli algoritmi *policy-iteration* prevedono l'alternanza di una fase di *valutazione* della politica (*policy evaluation*), in cui il critico si occupa di valutare l'efficacia della politica corrente (ad esempio, risolvendo le equazioni di Bellman per i vari stati), e una fase di *miglioramento* della politica (*policy improvement*), in cui l'attore si occupa di aggiornare la politica (ad esempio in modo *greedy*, cioè scegliendo in ogni stato l'azione localmente migliore).

Gli algoritmi *value-iteration*, invece, non usano una rappresentazione esplicita della politica: possono essere interpretati come una versione degenere di actor-critic in cui esiste solo il critico, che si occupa di stimare la funzione valore V o Q ottima.

Un'altra importante categoria è costituita dagli algoritmi *policy search*, che sfruttano una rappresentazione parametrica della politica per limitare lo spazio di ricerca. Anch'essi possono essere considerati una forma degenere di actor-critic, questa volta in cui esiste solo l'attore, che risolve il MDP cercando direttamente nello spazio delle politiche. Questi saranno i metodi che sfrutteremo in questa tesi.

## 2.3 Inverse Reinforcement Learning

Spesso, la funzione premio di un MDP può essere difficile da definire: non sempre, infatti, è immediato capire se e quanto scegliere una determinata azione in un determinato stato sia positivo per l'agente. Scopo dell'*apprendimento per rinforzo inverso* (*Inverse Reinforcement Learning*, *IRL*) è individuarla.

### 2.3.1 Introduzione

*Definizione di IRL*

Il problema di IRL è descrivibile (Russell, 1998) nella sua forma più generale in questi termini:

Dati:

- misurazioni del comportamento di un agente nel tempo, in diverse circostanze
- misurazioni degli input sensoriali dell'agente, se necessari
- un modello dell'ambiente, se disponibile

Trova:

- la funzione premio che l'agente punta ad ottimizzare

L'agente di riferimento è detto *esperto*; è anche possibile ottenere dimostrazioni da più esperti. In termini di politica, risolvere un problema di IRL significa prima codificare il comportamento dell'esperto in una politica $\pi^E$ di riferimento, e poi individuare una funzione premio tale che $\pi^E$ sia una politica ottima.

Questo approccio, il cui obiettivo è individuare la funzione premio, può fornire risultati migliori rispetto ad uno che si limita a costruire una politica tale che l'agente

imiti quella dell'esperto. Questo perché la funzione premio non descrive semplicemente il comportamento dell'agente, ma ne codifica le motivazioni in modo più approfondito: tale rappresentazione è più sintetica di una basata sulla politica, e permette di generalizzare il comportamento dell'agente anche a regioni dello spazio di stato non esplorate dall'esperto. Conoscendo la funzione premio, è anche possibile reagire a cambiamenti nel modello rigenerando una nuova politica ottimale a partire dalla funzione premio stessa, invece di doverla riapprendere completamente. Infine, esistono contesti in cui la funzione premio stessa è l'obiettivo della ricerca.

*Ambiguità del problema di IRL*

Tuttavia, il problema di IRL in forma stretta (cioè, individuare l'esatta funzione premio di un agente) è mal posto a causa dell'*ambiguità* intrinseca nelle soluzioni: infatti, è possibile che molte funzioni premio diverse risultino nello stesso comportamento. In alcuni casi, inoltre, alcune soluzioni possono risultare degeneri: ad esempio, una funzione premio costante – identicamente ($R(s,a) \equiv k \in \mathbb{R}$) o a causa di uno specifico assegnamento di pesi, ad esempio un vettore di pesi tutti nulli in caso di parametrizzazione lineare – significa che tutte le azioni sono ugualmente valide e, di conseguenza, qualsiasi politica è ottima. Il modo in cui questo problema di ambiguità è gestito dipende dallo specifico algoritmo.

### 2.3.2 Concetti fondamentali

*Parametrizzazione lineare*

Un approccio comune – introdotto in Ng e Russell (2000) – è supporre che la funzione premio sia formata dalla combinazione di più *funzioni base* (*basis functions*), in questo contesto anche denominate *feature* (caratteristiche), ognuna dei quali rappresenta uno degli obiettivi perseguiti dall'agente. Ad esempio, una funzione premio (così come una politica) può essere parametrizzata linearmente

$$R(\boldsymbol{s}, a) = \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{s}, a), \tag{2.7}$$

dove $\boldsymbol{\phi}(\boldsymbol{s}, a)$ è un vettore di feature, il cui valore dipende dallo stato in cui l'agente si trova e dall'azione che compie, e $\boldsymbol{\omega}$ è un vettore di *pesi*, i quali rappresentano le preferenze dell'agente per ciascuno degli obiettivi che le feature descrivono.

*Feature expectation*

A partire dalle feature si può introdurre il concetto di *feature expectation*, la cui più semplice definizione formale (Abbeel e Ng, 2004) è

$$\boldsymbol{\mu}(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(s_t, a_t)\right]. \tag{2.8}$$

Tale formula rappresenta il valore atteso accumulato da ciascuna feature $\phi$ seguendo la politica $\pi$ e tenendo conto del fattore di sconto $\gamma$.

### 2.3.3 Esempi di algoritmi IRL

Dalla sua definizione in (Russell, 1998), diversi algoritmi sono stati proposti per risolvere il problema di IRL. Un esempio di categorizzazione non netta né esaustiva, proposta in (Pirotta, 2016), è:

#### Algoritmi basati su feature-expectation

Gli algoritmi di questa famiglia utilizzano una parametrizzazione lineare della funzione premio, e di conseguenza della funzione valore. Essi puntano a minimizzare una

determinata metrica di dissomiglianza, basata sulla differenza tra la feature expectation della politica obiettivo e quella della politica dell'esperto ($\Delta\boldsymbol{\mu} = \boldsymbol{\mu}(\pi_{\boldsymbol{\omega}}) - \boldsymbol{\mu}(\pi^E)$, dove con $\pi_{\boldsymbol{\omega}}$ si indica che la politica trovata dipende dal vettore di pesi della funzione premio).

Esempi di algoritmi appartenenti a questa categoria sono:

- *Projection* (Abbeel e Ng, 2004), il primo algoritmo pratico di IRL presentato in letteratura. Ha come obiettivo risolvere un problema di norma $L_2$ quadrata, cioè usa come metrica il quadrato della norma euclidea della differenza tra feature expectation: $f(\boldsymbol{\omega}) = \|\Delta\boldsymbol{\mu}\|_2^2$.
- *MWAL* (*Multiplicative Weights for Apprenticeship Learning*) (Syed e Schapire, 2007). Ha come obiettivo individuare una politica le cui prestazioni sono buone al minimo quanto quelle dell'esperto ($J(\pi_{\boldsymbol{\omega}}) \geq J(\pi^E)$); data la scelta di parametrizzazione lineare ciò significa usare come metrica il vettore $\boldsymbol{\omega}$ dei parametri della funzione premio (tale che $\sum_i \omega_i = 1$ per risolvere il problema dell'ambiguità di IRL) moltiplicato per la differenza tra feature expectation: $f(\boldsymbol{\omega}) = \boldsymbol{\omega} \cdot \Delta\boldsymbol{\mu}$.

Entrambi hanno la limitazione di necessitare di un MDP, che ad ogni passo dell'algoritmo deve essere risolto con la parametrizzazione corrente per ottenere il valore di $\boldsymbol{\mu}(\pi)$.

Inoltre, in (Syed, Bowling e Schapire, 2008) è definito *LPAL* (*Linear Programming Apprenticeship Learning*), un algoritmo che integra il vincolo sulla qualità della politica trovata usato da MWAL nel contesto di ottimizzazione tramite programmazione lineare (LP). Tale algoritmo ha il vantaggio di poter essere risolto in un singolo passo di ottimizzazione; tuttavia impiegare la programmazione lineare vincola LPAL ad essere applicabile solo a MDP finiti e discreti, e rende onerosa la computazione su spazi stato/azione vasti.

#### Algoritmi supervisionati o strutturati

Gli algoritmi di questa famiglia riconducono il problema di IRL ad un problema di classificazione, quindi ad un problema standard di apprendimento supervisionato applicato sui dati grezzi o su una loro appropriata rielaborazione. In questi approcci, la politica è considerata un classificatore che assegna ad un input (lo stato) un'etichetta (l'azione); la qualità della politica è misurata su di una funzione punteggio (*score function*) basata sulla funzione valore. Ad esempio:

- *SCIRL (Structured Classification-based IRL)* (Klein et al., 2012) sceglie come politica quella che, in ogni stato, sceglie l'azione che massimizza la funzione $Q$ la quale, dato che si impiega una funzione premio parametrizzata linearmente, dipende linearmente dalla feature expectation ($Q(s,a) = \boldsymbol{\omega} \cdot \boldsymbol{\mu}(s,a)$, dove $\boldsymbol{\omega}$ è il vettore di parametri della funzione premio). L'algoritmo stima prima $\boldsymbol{\mu}(\pi^E)$ e poi, usando $Q$ come funzione punteggio, ottiene $\boldsymbol{\omega}$ e di conseguenza la funzione premio stessa.
- *CSI (Cascaded Supervised IRL)* (Klein et al., 2013), al contrario, non richiede una classe specifica di funzione premio. È articolato in due fasi, ognuna delle quali può sfruttare un qualsiasi algoritmo di classificazione: nella prima ricava la funzione $Q$; nella seconda ricava, a partire da essa e invertendo la relativa equazione di Bellman, la funzione premio.

Questi algoritmi (in versione euristica se l'esperto non è deterministico e sono note solo le traiettorie da esso prodotte) sono in grado di fornire risultati pressoché ottimali in condizioni opportune; tuttavia hanno la forte limitazione di non contemplare il caso in cui l'esperto non sia ottimo.

Una classe correlata di algoritmi, noti in letteratura come *maximal-margin-based*, riconduce il problema di IRL ad un problema di ottimizzazione vincolata (ad esempio, *Maximum Margin Planning* (MMP) (Ratliff, Bagnell e Zinkevich, 2006) usa la programmazione quadratica): il vincolo è che la politica dell'esperto sia migliore di ogni altra possibile politica di un certo margine; tuttavia tale requisito può essere rilassato aggiungendo variabili di scarto appropriate. È possibile interpretare questo approccio come un problema di classificazione, dove gli input sono delle istanze di MDP e le etichette sono le politiche ottime.

#### Algoritmi basati sulla verosimiglianza (*likelihood*)

Questi algoritmi puntano a stimare e successivamente ottimizzare la verosimiglianza del comportamento dell'esperto. Questo è ottenuto minimizzando la distanza – espressa in termini di divergenza di Kullback-Leibler o, equivalentemente, di entropia incrociata – tra il comportamento dell'esperto (espresso in termini di traiettorie o distribuzione stato/azione) e il comportamento dell'agente obiettivo.

A questa categoria appartengono:

- *Maximum Entropy IRL (MaxEnt IRL)* (Ziebart et al., 2008) punta a minimizzare la distanza tra la distribuzione delle traiettorie generate dall'esperto e una distribuzione della forma $e^{\boldsymbol{\omega}\cdot\boldsymbol{\mu}(\tau)}$ (dove $\boldsymbol{\mu}(\tau)$ è la feature expectation valutata sulla traiettoria $\tau$), considerando inoltre il vincolo di rispettare i premi ottenuti dalle azioni reali.

- *Relative Entropy IRL* (Boularias, Kober e Peters, 2011) punta a trovare i parametri tali da massimizzare le prestazioni dell'agente rispetto ad una funzione premio parametrica, e allo stesso tempo mantenere la distanza tra il suo comportamento e quello dell'esperto entro un certo limite. A differenza di MaxEnt IRL, questo approccio non richiede la conoscenza della funzione transizione.

#### Algoritmi basati su modelli probabilistici

Gli algoritmi di questa famiglia puntano a inferire tramite modelli probabilistici la politica o la funzione premio di un esperto a partire dalle sue traiettorie. Tale descrizione probabilistica permette di risolvere il problema dell'incertezza intrinseca del problema di IRL.

*Bayesian IRL* (Ramachandran e Amir, 2007) usa un approccio bayesiano comune alla maggioranza degli algoritmi di questa classe: la distribuzione a posteriori dei premi è ottenuta combinando una distribuzione a priori ($P(R(s,a)=\overline{r})$) e un modello probabilistico delle azioni dell'esperto data la funzione premio ($P(\{(s_1,a_1),\ldots,(s_n,a_n)\}|R)$). L'algoritmo suppone che la politica dell'esperto sia effettivamente volta ad ottimizzare una funzione premio (quindi, che non sia una politica esplorativa) e che tale politica sia stazionaria (cioè, invariante nel tempo).

In (Dimitrakakis e Rothkopf, 2012) è presentata un'estensione di questo metodo al contesto in cui è possibile che le traiettorie di esempio abbiano obiettivi (e quindi, corrispondano a politiche e funzioni premio) differenti: si punta quindi a stimare politica e

funzione premio di ogni traiettoria. L'incertezza ad esse relativa è gestita supponendo una distribuzione $\eta$ a priori congiunta su entrambe, poi condizionata dalle traiettorie osservate ($\eta(\cdot\,|\boldsymbol{T})$, dove $\boldsymbol{T}$ è l'insieme delle traiettorie). Sono proposte due varianti di algoritmo, che differiscono nella dipendenza tra funzione premio e politica:

- *Multitask Reward-Policy prior* (*MRP*): definisce una distribuzione a priori prodotto su funzione premio e parametri della politica ($\eta(R, \Pi) = \eta(R)\eta(\Pi)$, con $R$ e $\Pi$ insiemi di funzioni premio e politiche rispettivamente). Questo permette di specificare una politica univoca sulla quale la probabilità delle traiettorie è ben definita.
- *Multitask Policy Optimality prior* (*MPO*): definisce delle distribuzioni a priori su politica e su *ottimalità* della politica. Da esse è possibile ottenere una distribuzione a posteriori di funzioni premio condizionata dalla politica.

Un'ulteriore estensione di quest'ultimo approccio è formulata in (Tossou e Dimitrakakis, 2013), in cui è analizzato il caso di ambienti con dinamiche ignote.

### 2.3.4 Gradient Inverse Reinforcement Learning (GIRL)

*Gradient Inverse Reinforcement Learning* (GIRL) (Pirotta e Restelli, 2016) è un algoritmo IRL model-free che sfrutta il gradiente della politica dell'esperto per individuare una funzione premio che la renda ottima. È l'algoritmo impiegato nella parte pratica di questa tesi. Una sua caratteristica fondamentale è che il risultato può essere calcolato senza la necessità di risolvere un problema di RL diretto prima di passare alla fase di IRL.

#### Principi base

Si suppone di avere una classe di politica $\pi_{\boldsymbol{\theta}}(\boldsymbol{s}, a)$ e una classe di funzione premio $R_{\boldsymbol{\omega}}(\boldsymbol{s}, a)$, dipendenti rispettivamente dai vettori di parametri $\boldsymbol{\theta}, \boldsymbol{\omega} \in \mathbb{R}$. L'algoritmo non richiede una classe di parametrizzazione specifica; tuttavia la politica deve essere descrivibile in forma parametrica, differenziabile (essendo necessario il suo gradiente) e stocastica (o riconducibile a tale).

La cifra di merito usata per valutare la politica è il suo premio complessivo atteso, tenendo conto del fattore di riduzione (*expected discounted reward*):

$$J_{\boldsymbol{\omega}}(\pi_{\boldsymbol{\theta}}) = \int_S d_\mu^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}) \int_A \pi_{\boldsymbol{\theta}}(\boldsymbol{s}, a) R_{\boldsymbol{\omega}}(\boldsymbol{s}, a) da d\boldsymbol{s} \tag{2.9}$$

dove $d_\mu^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s})$ rappresenta la probabilità di trovarsi nello stato $s$ seguendo la politica $\pi$ quando il sistema è a regime, tenendo conto del fattore di riduzione $\gamma$ (Sutton et al., 2000).

Tale cifra di merito può essere riscritta in termini di funzione Q, e se ne può calcolare il gradiente rispetto a $\boldsymbol{\theta}$, cioè

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\omega}}(\pi_{\boldsymbol{\theta}}) = \int_S d_\mu^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}) \int_A \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{s}, a)_{\boldsymbol{\theta}} Q_{\boldsymbol{\omega}}^{\pi_{\boldsymbol{\theta}}}(s, a) da d\boldsymbol{s}.$$

Se non tutti i parametri di questa equazione sono noti, è possibile utilizzare delle loro stime. Ad esempio, se i parametri esatti della politica dell'esperto non sono

disponibili è possibile ricavarne un'approssimazione con una stima a massima verosimiglianza. Inoltre, è possibile stimare il gradiente utilizzando le traiettorie percorse dall'esperto tramite un qualunque metodo policy-gradient, ad esempio GPOMDP (Baxter e Bartlett, 2001).

Se $J_{\boldsymbol{\omega}}(\pi_{\boldsymbol{\theta}})$ è differenziabile rispetto al vettore di parametri $\boldsymbol{\theta}$, e la politica dell'esperto $\pi^E_{\boldsymbol{\theta}}$ è ottima data la funzione premio con pesi $\boldsymbol{\omega}$, allora il gradiente $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\omega}}(\pi^E_{\boldsymbol{\theta}})$ è nullo: la politica scelta è quindi un *punto stazionario* di $J_{\boldsymbol{\omega}}(\pi)$.

Affinché ciò sia vero, è sufficiente trovare un vettore $\boldsymbol{\omega}^*$ tale che

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \frac{1}{y} \left\| \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\omega}}(\pi^E_{\boldsymbol{\theta}}) \right\|_x^y \quad \forall x, y \geq 1,$$

cioè si può usare una qualsiasi funzione obiettivo basata sulla norma del gradiente. È importante notare che la scelta di tale categoria porta ad una funzione sempre convessa se la classe di funzione premio – la cui scelta è libera – è a sua volta convessa: si può quindi impiegare un qualsiasi metodo di ottimizzazione convessa per individuare $\boldsymbol{\omega}$.

Se l'esperto è ottimo, l'equazione precedente è verificata per il suo vettore di pesi $\boldsymbol{\omega}^E$. Se, al contrario, non è ottimo per qualsiasi motivo (ad esempio: la sua funzione premio non è rappresentabile dalla classe di funzioni premio scelta, le traiettorie a disposizione non sono sufficienti a stimare la sua politica con precisione sufficiente, o semplicemente la sua politica è subottimale), il vettore $\boldsymbol{\omega}^*$ trovato da GIRL è comunque la miglior approssimazione possibile di quello reale per la classe di funzioni premio scelta. Tale vettore, infatti, è quello che minimizza la norma del gradiente della politica, cioè quello che minimizza la variazione dei parametri della politica stessa, e si può comunque considerare affidabile se tale norma è piccola a sufficienza (il limite esatto dipende dall'applicazione).

#### Parametrizzazione lineare

Se si sceglie una funzione premio parametrizzata linearmente (eq. 2.7), si ha:

$$J_{\boldsymbol{\omega}}(\pi_{\boldsymbol{\theta}}) = \sum_i \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \omega_i \phi_i(s_t, a_t) \right] = \sum_i \omega_i \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \phi_i(s_t, a_t) \right] = \sum_i \omega_i \mu_i(\pi),$$

dove ogni $\mu_i(\pi)$ è la feature expectation (eq. 2.8) di un singolo obiettivo tra quelli che l'agente punta a perseguire (sia $\boldsymbol{J}(\pi) = [\mu_1, \ldots, \mu_n]$), e $\omega_i$ è il peso corrispondente tra quelli della funzione premio.

La cifra di merito $J_{\boldsymbol{\omega}}(\pi_{\boldsymbol{\theta}})$ è quindi espressa come *somma pesata* di obiettivi. Si può quindi interpretare l'algoritmo come la ricerca del vettore $\boldsymbol{\omega}$ che rende l'esperto paretiano.[2]

È possibile (Brown e Smith, 2005) formalizzare la condizione di ottimalità secondo Pareto di un vettore soluzione $\boldsymbol{\alpha}$ come l'appartenenza di $\boldsymbol{\alpha}$ allo *spazio nullo* (*null space*) o *nucleo* (*kernel*) di $D_{\boldsymbol{\theta}} \boldsymbol{J}(\pi)$:

$$\boldsymbol{\alpha} \in \ker(D_{\boldsymbol{\theta}} \boldsymbol{J}(\pi)) \quad \equiv \quad D_{\boldsymbol{\theta}} \boldsymbol{J}(\pi) \boldsymbol{\alpha} = \mathbf{0},$$

[2] Si ricorda che un vettore soluzione si dice ottimale secondo Pareto – o paretiano – se non esistono altre soluzioni migliori sotto qualche aspetto senza essere peggiori sotto altri (Hwang e Masud, 1979).

dove $D_{\boldsymbol{\theta}}\boldsymbol{J}(\pi)$ è la matrice jacobiana di $\boldsymbol{J}(\pi)$, cioè

$$D_{\boldsymbol{\theta}}\boldsymbol{J}(\pi) = \left[\nabla_{\boldsymbol{\theta}}\mu_1(\pi), \cdots, \nabla_{\boldsymbol{\theta}}\mu_n(\pi)\right].$$

In particolare, se ci si restringe al caso di combinazione convessa – cioè si aggiungono i vincoli $\alpha_i \geq 0$ e $\sum_i \alpha_i = 1\ \forall i$ – una soluzione paretiana è anche *stazionaria secondo Pareto* (Désidéri, 2012). Geometricamente, ciò significa che i vettori gradiente relativi ai singoli obiettivi sono – almeno nel caso ottimo – coplanari e disgiunti.

Questo risultato può essere sfruttato per risolvere il problema dell'ambiguità (§ 2.3.1): $\boldsymbol{\omega}$ può, senza perdita di generalità, essere ristretto al caso $\sum_i \omega_i = 1 \wedge \forall i : \omega_i \geq 0$. Di conseguenza,

$$\|\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\omega}}(\pi_{\boldsymbol{\theta}})\|_x = \left\| \left( \int_S d_\mu^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}) \int_A \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{s}, a) \boldsymbol{J}^\pi(\boldsymbol{s}, a) da d\boldsymbol{s} \right) \boldsymbol{\omega} \right\|_x = \|D_\theta \boldsymbol{J}(\pi)\boldsymbol{\omega}\|_x \forall x \in \mathbb{R}.$$

Se l'esperto è ottimo, il vettore soluzione $\omega$ appartiene effettivamente allo spazio nullo della matrice jacobiana; altrimenti tale soluzione rappresenta comunque la funzione premio che minimizza la variazione della politica.

### Plane GIRL

La scelta di una funzione premio parametrizzata linarmente permette di definire una versione di GIRL particolarmente efficiente, denominata *Plane GIRL* (PGIRL), in cui il problema è risolvibile in forma chiusa e in un singolo passo.

In condizioni di ottimalità, il vettore $\boldsymbol{\omega}$ è ortogonale all'iperpiano tangente alla frontiera di Pareto. È quindi possibile ricavarlo a partire da tale iperpiano: esso può essere identificato dai punti associati alla *matrice di Gram* di $D_{\boldsymbol{\theta}}\boldsymbol{J}(\pi)$, cioè a

$$\boldsymbol{G} = \left(D_{\boldsymbol{\theta}}\boldsymbol{J}(\pi)\right)^T D_{\boldsymbol{\theta}}\boldsymbol{J}(\pi).$$

Se tale matrice ha rango massimo, i punti ad essa associati identificano univocamente l'iperpiano: essendo $\boldsymbol{\omega}$ ad esso ortogonale, $\boldsymbol{\omega} \in \ker(\boldsymbol{G})$.

*Gestione delle degenerazioni*

Casi degeneri quali feature costanti, duplicate, linearmente dipendenti tra loro o inutilizzate possono essere eliminati incorporando una fase di pre-processing delle feature. In PGIRL, questo è fattibile verificando il rango della matrice jacobiana: se il numero di parametri della funzione premio è inferiore al numero di parametri della politica, un rango non massimo della matrice jacobiana indica una dipendenza tra obiettivi che è possibile risolvere rimuovendo le colonne interessate (tale soluzione equivale ad assegnare peso 0 ai relativi obiettivi). È importante notare che tale riduzione non è possibile se il numero di feature della funzione premio è superiore al numero di feature della politica: in tal caso, infatti, un rango inferiore al massimo possibile indica che la politica non è in grado di influenzare tutti gli obiettivi considerati dalla funzione premio, il che impedisce di individuare una soluzione unica. Politica e funzione premio dovrebbero quindi essere progettate in modo tale da evitare questa eventualità.

## 2.4 Clustering

La maggioranza delle tecniche finora descritte presuppongono che tutte le dimostrazioni di riferimento abbiano lo stesso obiettivo. Tuttavia, in casi reali può essere utile supporre che dimostrazioni differenti abbiano obiettivi differenti: applicando questa distinzione agli agenti diventa possibile suddividerli in categorie.

Questo è essenzialmente un problema di *clustering* (Bishop, 2006): con clustering si denota un insieme di tecniche di apprendimento non supervisionato il cui scopo è, dato un insieme di elementi con determinate caratteristiche, individuare in tale insieme delle categorie (*cluster*) non note a priori.

Proprio a causa della mancanza di informazioni a priori non è sempre immediato determinare la qualità di un determinato clustering: in generale, un clustering è considerato buono se, data una metrica di somiglianza (ad esempio, la semplice distanza euclidea), gli elementi appartenenti ad uno stesso cluster sono quanto più possibile simili tra loro e dissimili da quelli di altri cluster.

### 2.4.1 Algoritmi di clustering

Esistono diversi algoritmi di clustering, categorizzabili sulla base delle loro caratteristiche. Tra i più comunemente usati citiamo:

**k-means** (MacQueen, 1967) Si tratta di un semplice algoritmo a partizione, che separa gli elementi in modo netto. Scelto un numero K di cluster:

1. Gli elementi sono spartiti arbitrariamente tra i K cluster.
2. Per ogni cluster è calcolato il *centroide*, cioè la media di tutti i suoi punti.
3. Ogni elemento è riassegnato al cluster corrispondente al centroide più vicino.
4. I punti 2 e 3 sono ripetuti fino a convergenza, ad esempio finché il numero di elementi che cambiano cluster tra un'iterazione e l'altra non scende entro una certa soglia.

Per quanto applicabile in diversi contesti, k-means ha alcune importanti limitazioni: ad esempio, fatica a riconoscere cluster con densità molto diverse tra loro e cluster con forme non globulari.

**Expectation-Maximization (EM)** (Dempster, Laird e Rubin, 1977) Concettualmente simile a k-means, come esso suppone un numero prefissato di cluster, ma descrive l'appartenenza a tali cluster in termini di distribuzione di probabilità, cioè ogni elemento può essere assegnato contemporaneamente a più di un cluster con probabilità differenti: EM può quindi essere utilizzato per ottenere cluster *sfocati* (*fuzzy*). L'algoritmo opera in due fasi che si alternano ciclicamente fino a convergenza:

1. *Expectation*: assegna ogni esempio ad ogni cluster con una determinata probabilità, sulla base di dei parametri fissati.
2. *Maximization*: individua i parametri che massimizzano la verosimiglianza dell'assegnamento corrente.

È importante notare che entrambi gli algoritmi citati sono non deterministici, in quanto assegnazioni diverse dei parametri iniziali possono portare a cluster finali diversi.

### 2.4.2 Applicazioni a IRL nel caso multiutente

Un modo semplice ed inituitivo di combinare clustering e IRL è, data una classe di funzione premio comune, risolvere il problema di IRL per tutti gli esperti separatamente ed effettuare un clustering sui parametri ottenuti. Questo è il metodo che abbiamo impiegato in questa ricerca.

Tuttavia, esistono algoritmi più raffinati che combinano la fase di clustering con quella di calcolo della funzione premio. Tra essi citiamo:

#### Clustering EM per intenzioni multiple

Questo algoritmo (Babeş-Vroman et al., 2011) categorizza le traiettorie di esempio applicando un clustering di tipo EM. Una sua novità importante è che ipotizza che le traiettorie possano essere motivate da obiettivi (quindi, da funzioni premio) differenti, e non necessariamente dallo stesso obiettivo perseguito con grado variabile di affidabilità.

Questo specifico algoritmo utilizza una parametrizzazione lineare della funzione premio (eq. 2.7) e una politica stocastica $\pi_{\boldsymbol{\omega}}(\boldsymbol{s}, a)$ basata su una funzione Q esplorativa (si rimanda all'articolo per approfondimenti). Ogni cluster è caratterizzato da una probabilità a priori $\rho$ che un elemento vi appartenga e da una propria funzione premio definita da un vettore di pesi $\boldsymbol{\omega}$. Tali parametri sono inizializzati in modo arbitrario, e stimati con un processo iterativo:

1. fase Expectation:
   per ogni traiettoria $\tau_i$ calcola la probabilità che appartenga al cluster $j$:

   $$p_{ij}^t = P(\tau_i|\boldsymbol{\omega}_j^t) = \frac{1}{Z} \prod_{(\boldsymbol{s},a)\in\tau_i} \pi_{\boldsymbol{\omega}_j^t}(\boldsymbol{s}, a)\rho_j^t,$$

   dove l'apice $t$ indica che i valori si riferiscono all'iterazione corrente e $Z$ è un fattore di normalizzazione.

2. fase Maximization: ottimizza i parametri di ogni cluster:
   a) aggiorna le probabilità a priori $\rho$ come media campionaria degli assegnamenti di tutte le traiettorie:

   $$\rho_c^{t+1} = \frac{1}{N} \sum_{i=1}^{N} p_{ik}^t.$$

   b) individua il vettore di pesi $\boldsymbol{\omega}_c$ tale da massimizzare la verosimiglianza che ogni traiettoria si trovi nel cluster a cui è stata assegnata (nell'algoritmo presentato è utilizzato un algoritmo ad hoc chiamato *Maximum Likelihood IRL*, *MLIRL*):

   $$\boldsymbol{\omega}_c^{t+1} = \arg\max_{\boldsymbol{\omega}} \sum_{i=1}^{N} p_{ic} \log(P(\tau_i|\boldsymbol{\omega}_c^t)).$$

### Dirichlet Mixture Process – Bayesian IRL

*Dirichlet Mixture Process – Bayesian IRL* (*DMP-BIRL*) (Choi e Kim, 2012) è un'estensione al caso di agenti multipli degli algoritmi probabilistici citati in § 2.3.3. Esso combina l'approccio bayesiano di base di (Ramachandran e Amir, 2007) con il concetto di processo misto di Dirichlet per considerare la possibilità che le traiettorie si riferiscano a intenzioni distinte. Tale modello permette di avere un numero infinito numerabile di cluster: quindi, questo approccio ha rispetto a EM il vantaggio di non imporre a priori un numero fisso di cluster. Inoltre, dato che si basa su distribuzioni di probabilità, tali distribuzioni possono essere scelte in modo tale da codificare preferenze anche dovute al dominio applicativo. Altre caratteristiche utili di questo approccio sono il non essere parametrico e il poter classificare a posteriori nuove traiettorie in modo intuitivo.

L'algoritmo effettivo DMP-BIRL si basa su un sampler Metropolis-Hastings. A partire da una stima iniziale di assegnamento ai cluster $\boldsymbol{c}$ e funzioni premio $r_i$, alterna due passi fino al raggiungimento di un limite massimo di iterazioni:

1. Aggiorna gli assegnamenti ai cluster. Per ogni traiettoria:
   a) genera un assegnamento ad un cluster, basandosi sugli assegnamenti passati e su una distribuzione di Dirichlet.
   b) se il cluster è nuovo, ne genera la funzione premio da una distribuzione a priori apposita.
   c) assegna la traiettoria al cluster trovato con una certa probabilità minima.
2. Raffina le funzioni premio associate ai cluster trovati. Per ogni cluster:
   a) genera una nuova funzione premio a partire dalla vecchia, considerando anche il gradiente della sua distribuzione a posteriori e un errore normale.
   b) accetta la nuova funzione premio con una certa probabilità minima.

Si rimanda all'articolo originale per una formulazione più dettagliata.

# 3 Contesto di lavoro: consumo idrico di utenti domestici

Questo capitolo introduce il contesto di lavoro a cui il nostro metodo è stato applicato, cioè la modellizzazione del consumo idrico di utenti residenziali. È inoltre presentato brevemente come termine di paragone uno studio basato sui nostri stessi dati.

## 3.1 Modellizzazione del consumo idrico

Il processo di analisi dei consumi idrici di una popolazione si può riassumere in 4 fasi[1]:

1. Raccolta dei dati.
2. Eventuale disaggregazione tra utilizzi finali.
3. Modellizzazione degli utenti.
4. Definizione di WDMS (*water demand management strategies*, strategie di gestione della domanda idrica) eventualmente personalizzate.

### 3.1.1 Raccolta dei dati

È possibile raccogliere i dati di consumo a risoluzioni diverse, a seconda del tipo di analisi che si vorrà effettuare.

Una raccolta a *bassa risoluzione* è effettuata a livello cittadino o regionale, con una risoluzione temporale dell'ordine di alcuni mesi e una precisione di migliaia di litri ($m^3$). Dati di questo tipo sono più indicati ad analisi e pianificazioni di alto livello, data la loro impossibilità di rappresentare adeguatamente l'eterogeneità dei consumatori.

Una raccolta ad *alta risoluzione*, al contrario, è effettuata mediante rilevatori intelligenti (*smart meter*) di vario tipo (ad esempio, misuratori di flusso e sensori di pressione) a livello della singola abitazione (monitoraggio *non invasivo*), o addirittura della singola apparecchiatura (*invasivo*), con risoluzione sia temporale che di consumo molto più fine, anche fino a pochi secondi e meno di un litro rispettivamente. Il metodo invasivo è tuttavia difficile da applicare in pratica proprio a causa dell'elevato numero di installazioni necessarie, che portano a costi elevati e al rischio di minore accettazione da parte degli utenti.

Utili all'analisi dei consumi sono inoltre i *dati psicografici* degli utenti in questione, cioè il numero di abitanti nella residenza considerata, le loro abitudini, caratteristiche dell'abitazione e simili. Tuttavia, non sempre tali dati sono disponibili a causa dell'elevata quantità di risorse necessarie alla loro raccolta.

[1] Per una trattazione più approfondita, completa di riferimenti bibliografici dettagliati, si rimanda a (Cominola et al., 2015) e (Moro e Riva, 2016), su cui questa sezione è basata.

### 3.1.2 Disaggregazione

Questa fase non è necessaria in caso di monitoraggio invasivo.

Gli approcci esistenti possono essere categorizzati in algoritmi basati su *alberi decisionali* e algoritmi basati su tecniche di *apprendimento automatico* (*machine learning*, ML).

In generale, questa fase è difficile da eseguire: finora sono stati sviluppati pochi algoritmi accurati, e se anche la loro precisione è abbastanza alta (dell'ordine del 70-94% a seconda del metodo) richiedono una grande quantità di informazioni non reperibili in modo automatico, a volte un numero elevato di installazioni di sensori e, più in generale, un grande impegno diretto da parte degli analisti.

### 3.1.3 Modellizzazione degli utenti

A seconda della risoluzione dei dati scelta, è possibile modellizzare gli utenti sia a livello regionale che a livello locale; in particolare, il secondo approccio è volto a rappresentare i consumi delle singole abitazioni, preservando quindi l'eterogeneità dei diversi tipi di consumatori.

Gli approcci esistenti possono essere raggruppati in due categorie:

- I *modelli descrittivi* si limitano ad analizzare i dati di consumo degli utenti. Possono essere utilizzati per, ad esempio, costruire a partire dalle tendenze storiche profili utenti che possono essere utilizzati come riferimento per individuare possibili miglioramenti. Tuttavia, non sono in grado di prevedere autonomamente l'efficacia di potenziali nuove WDMS.

- I *modelli predittivi*, al contrario, cercano di stimare attivamente la domanda idrica delle abitazioni. L'analisi può essere effettuata in due fasi:

  1. *analisi multivariata*: identifica i *driver* (motivazioni) principali dei consumi a partire da un elenco di candidati appartenenti a varie categorie (economici, climatici, psicografici), mediante tecniche di data mining e/o considerazioni empiriche.
  2. *modellizzazione comportamentale*: costruisce un modello matematico che descrive quantitativamente il consumo dell'utente in funzione dei driver identificati in precedenza. Tali modelli possono essere costruiti per singoli utenti o per reti sociali di utenti: il secondo approccio, basato sulla teoria dei sistemi multiagente, è più oneroso ma – essendo in grado di rappresentare le interazioni tra utenti – può portare a risultati più realistici.

### 3.1.4 Definizione di WDMS personalizzate

Le politiche di gestione possono essere categorizzate (Inman e Jeffrey, 2006) in:

- *tecnologiche*: consistono nell'installazione di impianti più efficienti.

- *finanziarie*: consistono in un controllo delle tariffe dipendente dall'elasticità della domanda idrica.

- *legislative*: consistono in ordinanze e limitazioni volte a ridurre i consumi, in particolare in caso di siccità.

- *manutentive*: consistono nella manutenzione degli impianti per riparare e prevenire perdite nella rete idrica, che possono costituire un rilevante fattore di spreco.
- *educative*: consistono nell'istruire gli utenti al risparmio tramite campagne di sensibilizzazione.

Gli studi in questo campo suggeriscono che le politiche di natura impositoria siano potenzialmente efficaci a breve termine, più che a lungo termine: ad esempio, gli utenti possono reagire ad un aumento di prezzo diminuendo i propri consumi, ma tali consumi tendono a riassestarsi al valore iniziale con il passare del tempo; i divieti possono inoltre essere accettati con difficoltà dalla comunità. Le politiche educative, invece, funzionano in modo opposto: le riduzioni iniziali sono esigue, ma se si riesce ad incentivare un comportamento più attento da parte degli utenti (ad esempio, comunicando loro quanto hanno risparmiato) il risparmio a lungo termine può essere ben più elevato.

## 3.2 Analisi mediante eigenbehavior

Presentiamo ora uno studio affine al nostro, il cui scopo è identificare le abitudini degli utenti.

### 3.2.1 Introduzione

Gli *eigenbehavior* ("autocomportamenti") (Eagle e Pentland, 2009) sono rappresentazioni dei componenti principali del comportamento di un utente.

Formalmente, siano $\Gamma_i$ un vettore di comportamenti di dimensione $H$, $\Psi = \frac{1}{D}\sum_{n=1}^{D}\Gamma_n$ il vettore comportamento medio e $\Phi_i = \Gamma_i - \Psi$ la deviazione di ogni vettore dalla media. A partire da queste quantità è possibile definire una *matrice di covarianza*:

$$C = \frac{1}{H}\sum_{n=1}^{H}\Phi_n\Phi_n^T$$

i cui autovettori e autovalori sono, rispettivamente, gli eigenbehavior e i relativi pesi.

Un eigenbehavior è tanto più importante quanto la varianza da esso indotta nei dati – corrispondente al suo autovalore – è elevata: ciò significa che l'eigenbehavior rappresenta un comportamento particolarmente comune. È quindi possibile rappresentare il comportamento di un utente come la combinazione lineare dei suoi eigenbehavior con autovalori più alti: nel caso di studio di Eagle e Pentland (2009) – cioè un limitato tracciamento della posizione geografica di degli utenti lungo l'arco di una giornata – i 6 eigenbehavior principali si sono dimostrati sufficienti a spiegare i dati con una precisione minima del 90%.

È inoltre possibile effettuare la stessa analisi su gruppi di utenti: il comportamento di un utente può essere quindi descritto a partire dagli eigenbehavior del gruppo al quale appartiene.

### 3.2.2 Applicazione al consumo idrico

È possibile utilizzare gli eigenbehavior per modellizzare i consumi idrici. Come riferimento, presentiamo la sintesi di uno studio (Cominola et al., 2016) effettuato considerando gli stessi dati – la cui struttura sarà descritta più nel dettaglio in § 5.1 – su cui si basa questa tesi.

I consumi orari ($c$) sono stati suddivisi in 4 classi:

- consumi nulli (L0): $c = 0\ L/h$.
- consumi bassi (L1): $c \in (0, 12]\ L/h$.
- consumi medi (L2): $c \in (12, 100]\ L/h$.
- consumi elevati (L3): $c > 100\ L/h$.

A partire da tali classi, rappresentate come variabili binarie, sono stati estratti gli eigenbehavior. Un'analisi dei dati ha mostrato che il primo eigenbehavior permette di spiegare in media il 50% della varianza dei dati, 10 eigenbehavior permettono di spiegarne il 75% e 60 circa il 100%. Quindi, già il primo eigenbehavior è sufficiente a catturare gran parte della variabilità del comportamento degli utenti, e può essere sfruttato per una categorizzazione efficace.

I dati sono stati sottoposti ad un clustering con k-means (§ 2.4.1) basato sul primo eigenbehavior, suddividendo gli utenti nelle 3 categorie mostrate in Figura 3.1:

- la prima categoria dà elevata importanza ai consumi nulli e scarsa agli altri. Questo può significare che le abitazioni sono spesso disabitate, o hanno pochi abitanti e quindi poca necessità di consumare.
- la seconda categoria dà elevata importanza ai consumi nulli e medi rispettivamente nelle ore notturne e diurne, importanza media ai consumi bassi e presenta due picchi di consumie elevati di mattina e di sera. Si suppone che questa categoria, la più grande delle tre, rappresenti l'utente medio.
- la terza categoria dà poca importanza ai consumi nulli, soprattutto durante il giorno, e una prevalenza di consumi medi o alti, mantenendo la coppia di picchi di consumi elevati. È possibile che queste abitazioni siano simili a quelle della seconda categoria, ma con più abitanti.

La procedura è stata quindi in grado di fornire una buona descrizione delle abitudini degli utenti, i quali si sono dimostrati categorizzabili in modo significativo.

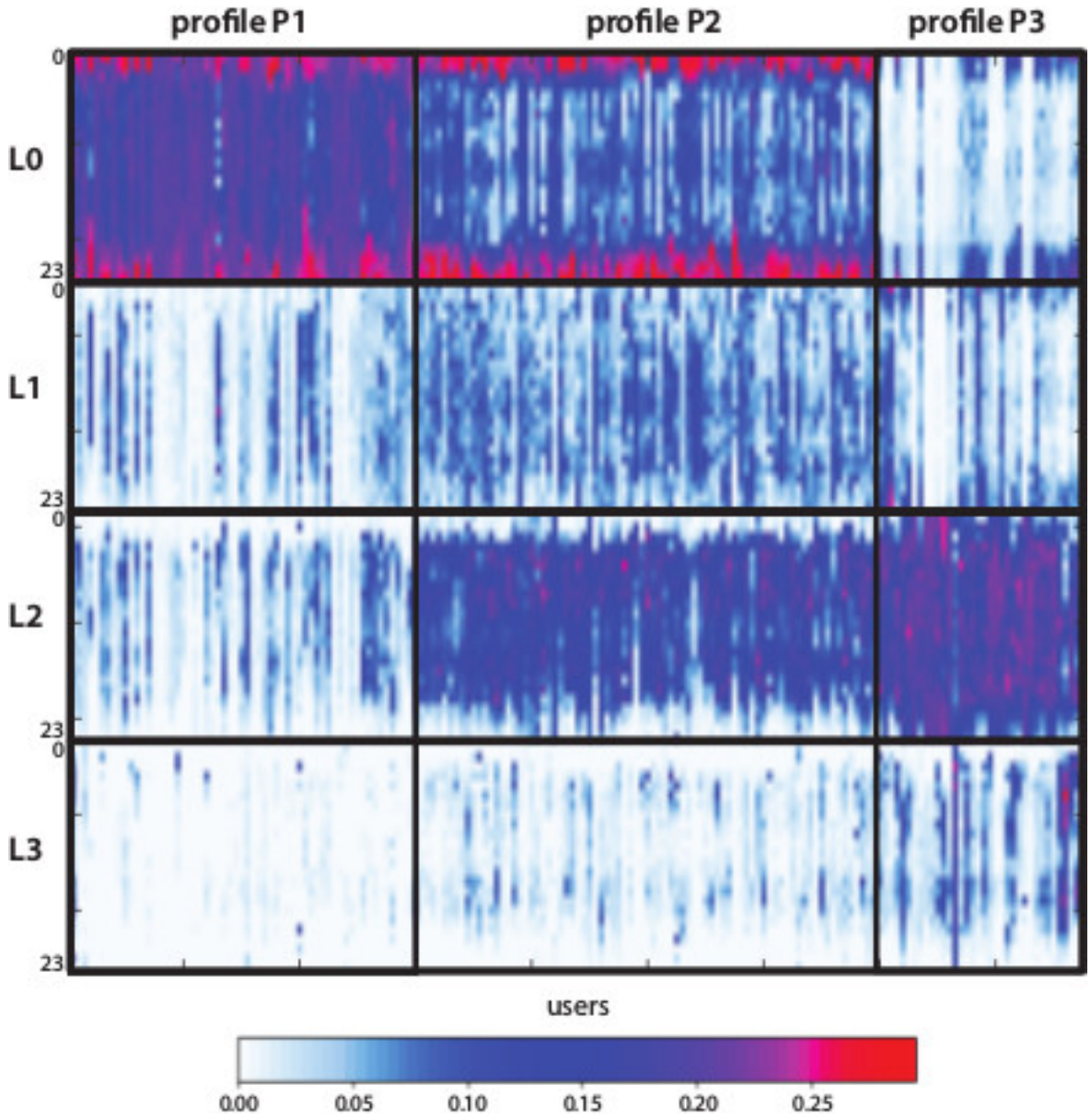



Figura 3.1: Risultati della categorizzazione di utenti mediante eigenbehaviors, provenienti da (Cominola et al., 2016). Ogni colonna di ogni gruppo è il primo eigenbehavior di un utente, e ogni riga corrisponde ad un'ora del giorno per ognuna delle 4 classi di consumo. Il colore dei punti ne indica il peso.

# 4 GIRL per agenti multipli

In questo capitolo è presentato un procedimento che estende GIRL al caso di agenti multipli, consistente nell'applicazione a posteriori di metodi di clustering sulle funzioni premio ottenute. Sono descritte ad alto livello le varie fasi del processo. Un caso di studio approfondito sarà presentato nel Capitolo 5.

## 4.1 Introduzione

L'algoritmo GIRL (§ 2.3.4) e i suoi derivati sono concepiti per ottenere una descrizione della funzione premio di un singolo esperto. Lo scopo di questa tesi è estendere le sue funzionalità al caso di agenti multipli, che si suppone agiscano in base a funzioni premio diverse.

Supponiamo però una classe di funzione premio comune: la nostra ipotesi è che esista un gruppo di obiettivi condivisi dalla maggioranza degli agenti che consideriamo, e le differenze tra tali agenti siano dovute a diversi gradi di preferenza verso obiettivi specifici. Tale ipotesi non vincola tutti gli agenti a dover per forza voler perseguire tutti gli obiettivi: se un agente non è interessato ad alcuni di essi, i parametri ad essi corrispondenti saranno idealmente nulli.

Una volta individuati i parametri di tutti gli agenti, è possibile categorizzare gli agenti con la semplice applicazione di un algoritmo di clustering sui parametri finali.

## 4.2 Fasi del processo

Il processo è strutturato in quattro fasi, sintetizzate in Figura 4.1:

1. Raccolta e analisi dei dati.
2. Definizione e calcolo delle politiche degli utenti.
3. Definizione e calcolo delle funzioni premio degli utenti.
4. Individuazione di categorie di utenti mediante clustering.

### 4.2.1 Raccolta e analisi dei dati

Alla base dell'intero procedimento si trovano i dati grezzi raccolti dagli agenti. È possibile che tali dati contengano imprecisioni o spurietà di vario tipo: può quindi essere opportuna una fase di pre-processing per individuare e scartare elementi problematici quali outlier o sequenze con una grande quantità di dati mancanti.

I dati raffinati in questo modo vengono poi analizzati per individuarne le caratteristiche fondamentali. A questa fase possono contribuire considerazioni a priori sul

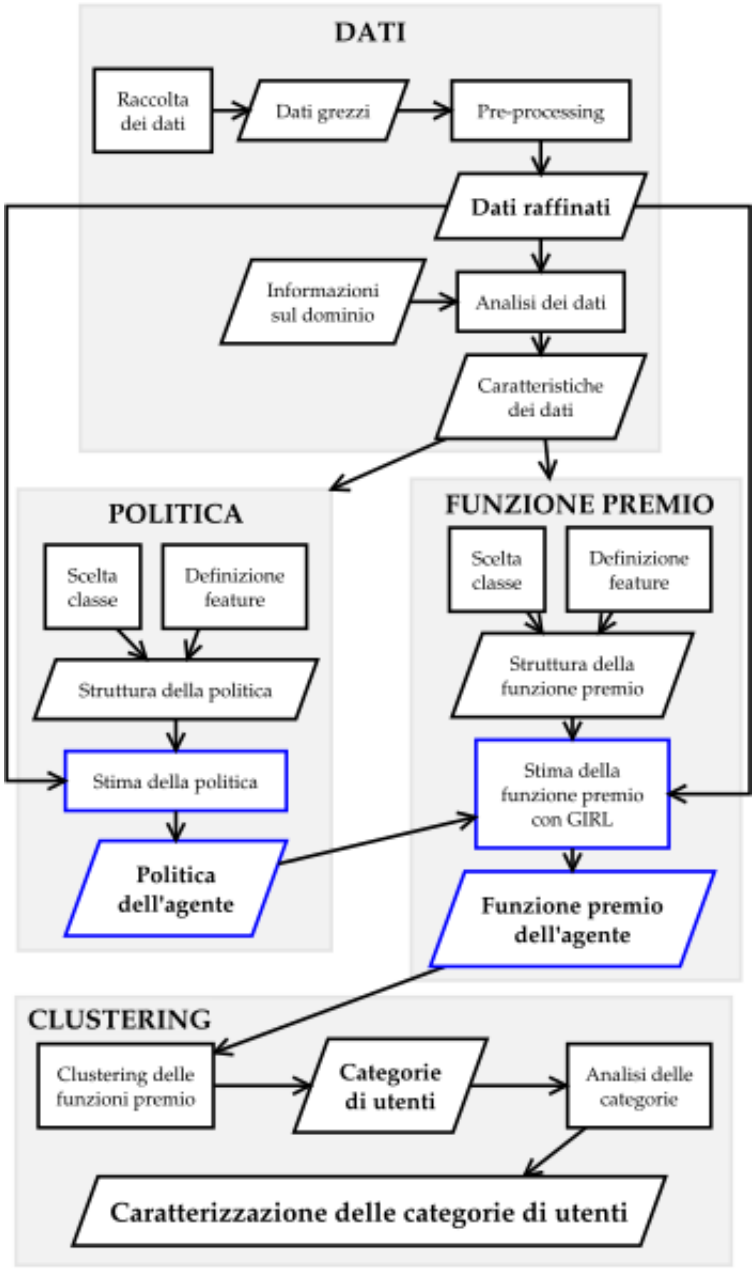



Figura 4.1: Sintesi del processo di caratterizzazione di più utenti combinando GIRL e clustering. I riquadri con bordo blu indicano le fasi che devono essere eseguite separatamente per tutti gli agenti.

dominio applicativo, provenienti da risultati di studi precedenti. La struttura così individuata fungerà da punto di partenza per definire una politica e una funzione premio atte a rappresentare in modo più efficace possibile il comportamento degli agenti.

### 4.2.2 Definizione della politica

La politica deve essere adeguata a descrivere le azioni compiute dagli agenti con un sufficiente grado di precisione. Data la nostra scelta di sfruttare GIRL, tale politica deve essere stocastica, parametrica e differenziabile rispetto a tali parametri: una scelta naturale, quindi, è scegliere come *classe di politica* una distribuzione di probabilità parametrica.

I parametri di tale distribuzione sono espressi come combinazione di *funzioni base* o *feature* dipendenti dallo stato: la struttura dello stato e di tali feature è definita sulla base delle considerazioni effettuate sui dati al punto precedente e su esperimenti pratici.

Una volta scelte classe di politica e feature, si procede alla *stima* dei parametri sulla base dei dati reali. In particolare, è possibile effettuare una classica stima *a massima verosimiglianza*, cioè che punta ad individuare i parametri tali da massimizzare la verosimiglianza dei dati osservati.

L'operazione di stima è eseguita su ogni agente separatamente. Se lo si ritiene opportuno, è possibile definire per ogni agente una classe di politica personalizzata.

### 4.2.3 Definizione della funzione premio

Come per la politica, vengono scelte una classe di funzione premio e delle feature adatte a descrivere il comportamento dell'agente il meglio possibile, e come per la politica alla funzione premio è associato un vettore di parametri. Tuttavia, è opportuno notare che, nonostante la somiglianza nelle rispettive fasi di progettazione, politica e funzione premio hanno obiettivi distinti: dove la politica punta a descrivere il meglio possibile le *azioni* compiute dall'agente, classe e funzioni base della funzione premio devono essere tali da permettere di distinguerne le *motivazioni*.

Al contrario della politica, non è possibile definire funzioni premio con strutture personalizzate, dato che, per poter distinguere gli agenti in categorie, i loro vettori di parametri devono essere comparabili tra loro. Nei nostri esperimenti abbiamo effettuato la scelta comune di una semplice parametrizzazione lineare (eq. 2.7).

Una volta definita la funzione premio globale, si procede – ancora separatamente per ogni agente – alla *stima* dei suoi parametri sulla base, oltre che dei dati reali, della politica parametrica ottenuta al punto precedente.

### 4.2.4 Clustering

Alcuni degli approcci presenti in letteratura che combinano IRL e clustering (§ 2.4.2) categorizzano le traiettorie compiute dagli agenti. Il nostro procedimento, al contrario, agisce con una granularità inferiore in quanto siamo interessati alle peculiarità degli agenti considerati nel proprio insieme: il clustering è quindi applicato a posteriori.

I vettori di pesi delle funzioni premio dei vari agenti ottenuti al punto precedente sono categorizzati per mezzo di un qualsiasi algoritmo di clustering (alcuni dei quali so-

no stati descritti in § 2.4.1). Essendo diversi algoritmi di clustering non deterministici, può essere necessario provare più assegnamenti per trovarne uno significativo.

Infine, le categorie così ottenute sono analizzate per verificare che le caratteristiche delle loro funzioni premio siano effettivamente indicative di proprietà reali degli agenti.

# 5 Caso di studio e risultati

Questo capitolo espone il lavoro da noi svolto seguendo la metodologia delineata nel capitolo 4. Per ognuna delle quattro fasi (raccolta e analisi dei dati, definizione della politica, definizione della funzione premio, clustering) sono presentate assunzioni, scelte implementative e risultati ottenuti.

## 5.1 Raccolta e analisi dei dati

### 5.1.1 Raccolta dei dati

Il dataset a nostra disposizione è costituito da dati di consumo residenziale ottenuti per mezzo di smart meter in abitazioni situate nella frazione di Tegna (comune di Terre di Pedemonte, distretto di Locarno, Canton Ticino, Svizzera) dalla Società Elettrica Sopracenerina nel contesto del progetto SmartH2O[1] (Rizzoli et al., 2014). Tali dati sono stati raccolti in 250 abitazioni nel periodo gennaio – novembre 2015.

Questi dati sono stati sottoposti ad una fase di pre-processing automatico per limitare i casi problematici quali letture mancanti, errori di misura, periodi di rodaggio, abitazioni prive o quasi di consumi in tutto l'intervallo considerato, e abitazioni i cui rilevamenti effettivi hanno avuto inizio diversi mesi dopo quelli delle altre. Il dataset finale[2] è costituito dai dati di consumo di 175 abitazioni limitati al periodo 30 marzo – 7 novembre 2015.

È stato registrato il consumo idrico complessivo per abitazione, misurato ad intervalli orari (ogni campione è la somma dei consumi effettuati nell'ultima ora) e preciso al litro; i dati sono espressi in metri cubi (quindi, 1 litro corrisponde ad un valore di 0,001). A tali consumi sono associate etichette temporali (giorno e ora), da noi corrette per tenere conto dell'ora legale. In totale, quindi, abbiamo a disposizione 5352 letture orarie per ogni abitazione, per un totale di 936600 letture complessive.

Abbiamo inoltre a disposizione dati meteorologici (temperatura e precipitazioni) relativi al periodo in esame, con risoluzione di 10 minuti e rispettivamente di 1 °C e 1 mm, dal database dell'Osservatorio Ambiente della Svizzera Italiana[3], riferiti però non a Tegna ma al vicino comune di Locarno.

Non sono presenti dati demografici, né distinzioni tra fonti di consumo (ad esempio bagni, giardino o lavandini), né altre informazioni specifiche sulla singola abitazione (ad esempio, il numero di bagni o la presenza di giardini). Tale assenza limita l'efficacia del nostro approccio: ad esempio, è lecito supporre che il numero di abitanti influenzi notevolmente i consumi e che un giardino sia una fonte notevole di consumo. Inol-

[1] `http://www.smarth2o-fp7.eu`

[2] È importante notare che non tutti i casi evidentemente problematici sono stati individuati ed esclusi: abbiamo deciso comunque di mantenerli per verificare il funzionamento del nostro metodo in simili casi limite.

[3] `http://www.oasi.ti.ch/web/dati/stazioni-di-rilevamento.html?domain=meteo&code=13`

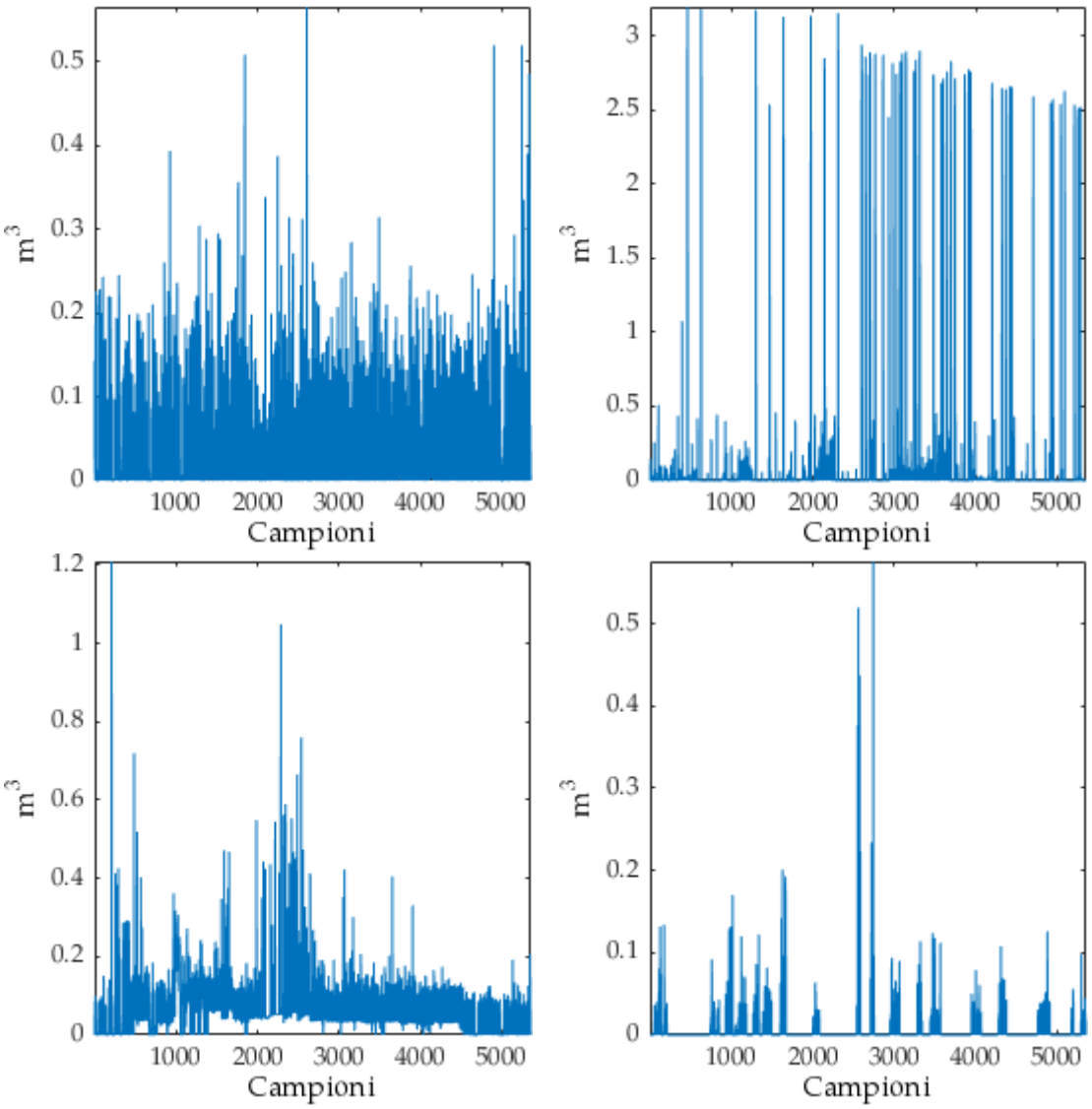



Figura 5.1: Esempi di profili di consumo complessivi. I grafici hanno scale differenti, per meglio rappresentare le peculiarità delle singole abitazioni.

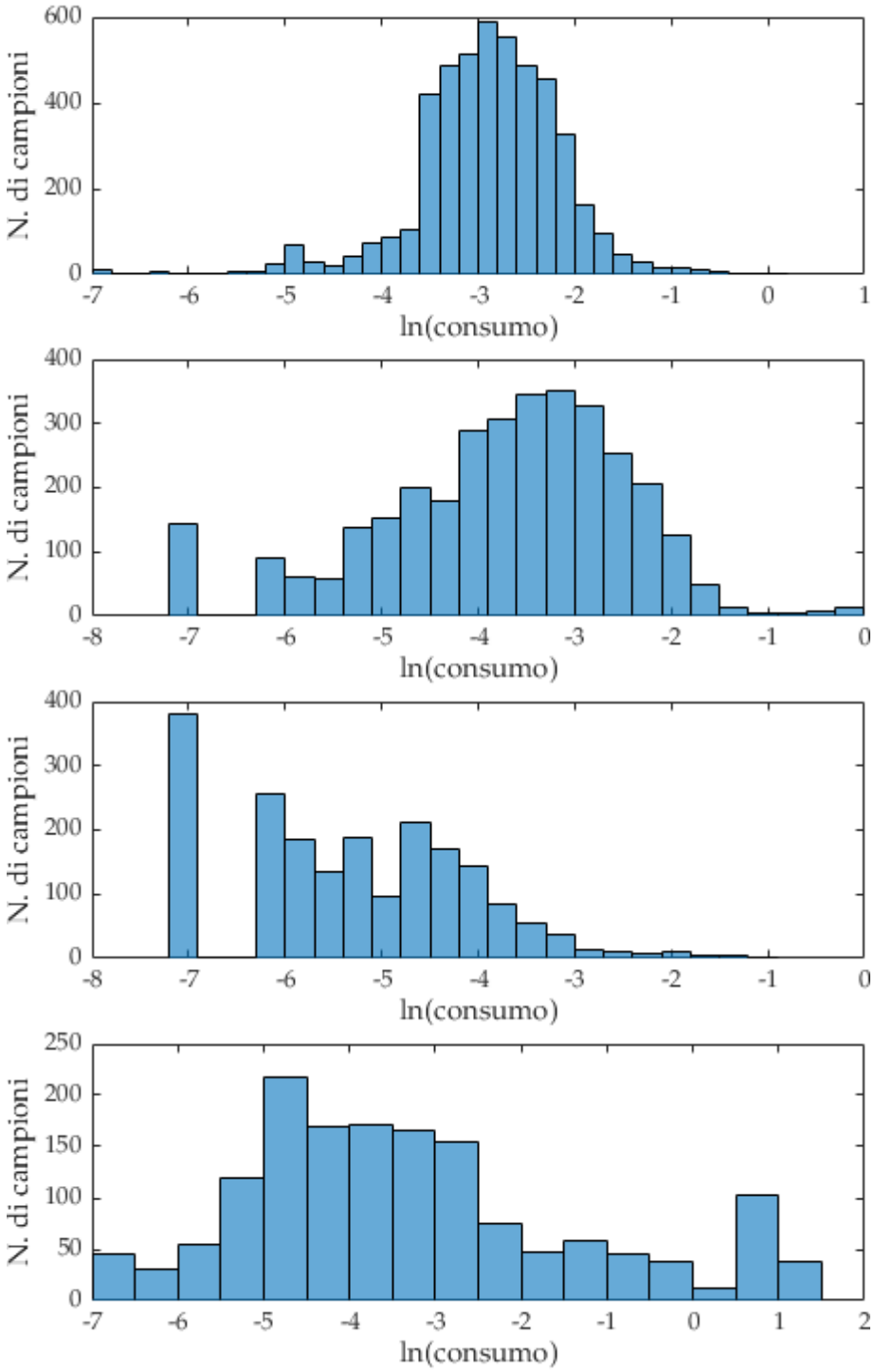



Figura 5.2: Istogrammi dei consumi reali di alcune abitazioni, escludendo i consumi nulli. È mostrato un esempio ben approssimabile da una distribuzione lognormale, insieme ad alcune discrepanze comuni.

tre, considerando l'onerosità intrinseca del processo e la mancanza di informazioni specifiche, non abbiamo ritenuto opportuno tentare una disaggregazione.

### 5.1.2 Analisi della struttura dei dati

I profili di consumo delle varie abitazioni sono molto eterogenei: alcune presentano consumi regolari, altre un numero elevato di picchi, diverse hanno periodi prolungati di consumo continuo e altre ancora intervallano sprazzi di consumo a periodi prolungati di consumi nulli. Alcuni esempi di profili globali sono forniti in Figura 5.1.

I consumi delle varie abitazioni sono spesso distribuiti in modo prevalentemente lognormale. Tuttavia, tale corrispondenza non è esatta: spesso si ha una coda di consumi molto bassi, che in alcuni casi domina la parte lognormale principale; inoltre alcune abitazioni presentano una coda di consumi elevati. Alcuni esempi di distribuzione dei consumi sono presentati in Figura 5.2.

## 5.2 Struttura delle traiettorie

Abbiamo deciso di considerare ogni abitazione separatamente, per preservarne le peculiarità.

Ogni singola ora costituisce una traiettoria a sé: pur supponendo una componente autoregressiva limitata nei consumi, non è immediato capire come suddividere in traiettorie più brevi l'unica serie continua di dati che abbiamo a disposizione per ogni abitazione. Inoltre, alcune transizioni (ad esempio, quelle dovute allo scorrere del tempo) non sono sotto il controllo dell'utente e quindi non abbiamo interesse a modellizzarle. È tuttavia lecito supporre che ogni consumo influenzi in qualche modo quelli delle ore successive: tale dipendenza è codificata nello stato, come spiegato nella sezione successiva.

Questa scelta ci permette inoltre di avere a disposizione un numero elevato di traiettorie per ogni abitazione.

## 5.3 Definizione della politica

L'azione effettuata dalla politica è, nel nostro caso, la quantità d'acqua consumata in una determinata ora.

### 5.3.1 Scelta della classe di politica

Per poter impiegare GIRL la politica deve essere stocastica (§ 2.3.4). Nel nostro caso, tale scelta appare ragionevole: difficilmente un utente, a parità di condizioni, consumerà la stessa esatta quantità di acqua; inoltre la quantità relativamente scarsa di informazioni a nostra disposizione aumenta la probabilità che più stati – la cui struttura sarà descritta in seguito – siano indistinguibili tra loro.

Considerata la distribuzione reale dei dati (Figura 5.2), una politica basata su una distribuzione lognormale appare una semplificazione ragionevole. Tuttavia, tale scelta non è da sola in grado di spiegare le numerose azioni nulle (circa il 46% del totale). Abbiamo quindi impiegato una politica combinata, composta da:

- una parte *bernoulliana*, il cui scopo è determinare se, in un determinato stato, l'agente consuma o meno una qualsiasi quantità di acqua[4]:

$$f(\boldsymbol{s}, a|p) = \begin{cases} p(\boldsymbol{s}) & \text{se } a > 0 \\ 1 - p(\boldsymbol{s}) & \text{se } a = 0 \end{cases}.$$

- una parte *lognormale*, il cui scopo è modellizzare il consumo dell'agente se tale consumo è positivo. Tale distribuzione usa, invece della deviazione standard, la *precisione*, cioè il suo reciproco[5] ($\alpha = \frac{1}{\sigma}$) per permettere una stima biconvessa dei parametri:

$$f(\boldsymbol{s}, a|\mu, \alpha) = \frac{\alpha(\boldsymbol{s})}{a\sqrt{2\pi}} \exp\Big(-\frac{(\ln a - \mu(\boldsymbol{s}))^2 \alpha(\boldsymbol{s})^2}{2}\Big).$$

Tutti i parametri della distribuzione congiunta dipendono dallo stesso vettore di feature, che descriveremo nel dettaglio nella sezione successiva. In particolare:

- il parametro della parte bernoulliana vi dipende in modo logistico:

$$p(\boldsymbol{s}) = \frac{1}{1 + \exp(-\boldsymbol{p}^T \boldsymbol{\phi}(\boldsymbol{s}))}.$$

- i parametri della parte lognormale vi dipendono linearmente:

$$\mu(\boldsymbol{s}) = \boldsymbol{\mu}^T \boldsymbol{\phi}(\boldsymbol{s})$$
$$\alpha(\boldsymbol{s}) = \boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{s}).$$

Il vettore gradiente della politica combinata è semplicemente l'unione dei vettori gradienti delle due sottopolitiche, dato che esse non condividono parametri. Tuttavia, il gradiente della parte lognormale non è definito per azioni nulle, dato che nella sua struttura l'azione corrente si trova a denominatore. Di conseguenza, quando si calcola il gradiente della politica combinata il gradiente della parte bernoulliana è calcolato normalmente, mentre quello della parte lognormale è calcolato supponendo un'azione pari a 1 litro (0,001), corrispondente alla minima azione possibile data la risoluzione dei dati.

### 5.3.2 Scelta delle feature della politica

Sulla base di considerazioni a priori sul nostro dominio e di ragionamenti intuitivi, abbiamo ipotizzato le seguenti categorie di feature:

- *consumi pregressi*, sia puntuali (ad esempio, il consumo dell'ora precedente o di 24 ore prima) che aggregati (ad esempio, il consumo complessivo delle ultime 24 ore). Questa scelta ci fornisce delle componenti autoregressive che abbiamo supposto influire sul consumo disincentivando consumi vicini nel tempo e rappresentando consumi periodici.

[4] Si ricorda che una distribuzione bernoulliana autentica ammette come uniche azioni possibili 0 (fallimento) e 1 (successo), quindi nel caso generico la prima condizione è $a = 1$.

[5] Tradizionalmente, con precisione si indica il reciproco della *varianza*; dato lo stretto legame tra varianza e deviazione standard definiremo comunque precisione l'inverso della deviazione standard per semplicità.

| Feature | Classifica | N. di abitazioni |
|---|---|---|
| C1cum | 1,42 | 147 |
| Time | 2,57 | 103 |
| C24lag | 2,99 | 69 |
| C48lag | 3,22 | 67 |
| C2cum | 3,65 | 62 |
| Day | 3,85 | 61 |
| C1settlag | 3,78 | 45 |
| C3cum | 4,96 | 23 |
| C24cum | 5,91 | 11 |
| C48cum | 5,6 | 5 |
| T1mean | 7,8 | 5 |
| C12cum | 5 | 4 |
| C1settcum | 6 | 2 |
| P12cum | 2 | 1 |
| T48mean | 7 | 1 |
| T12mean | 8 | 1 |
| C2settcum | 10 | 1 |

Tabella 5.1: Risultati della feature selection sulle singole abitazioni. *Classifica* è la posizione media in classifica si una feature quando è selezionata per un'abitazione. *CXlag* indica il consumo verificatosi X ore prima di quello in esame, *CXcum* il consumo cumulativo nelle ultime X ore, *TXmean* la temperatura media nelle ultime X ore, *PXcum* le precipitazioni cumulative nelle ultime X ore. Le barre laterali forniscono una rappresentazione visuale del numero di abitazioni. Le feature da noi scelte sono quelle al di sopra della linea orizzontale.

- *informazioni temporali*: giorno della settimana e ora del giorno. Abbiamo ipotizzato che i consumi in un giorno feriale siano diversi da quelli di un giorno festivo, e che alcuni orari siano propensi più di altri a generare consumi (ad esempio, le ore dei pasti saranno probabilmente ore di punta).
- *dati meteorologici*: temperatura media e precipitazioni cumulative in intervalli determinati. Abbiamo ipotizzato che il clima influisca, ad esempio, sui consumi dovuti all'igiene personale e alla manutenzione di un eventuale giardino.

Queste feature sono poi state analizzate mediante *feature selection*: tale operazione, nel nostro caso effettuata usando l'algoritmo *Extra-Trees* (Galelli e Castelletti, 2013), ha come obiettivo identificare le feature che influiscono maggiormente sull'azione – cioè, nel nostro caso, sul consumo ad ogni ora considerata – usando come metrica di valutazione il *coefficiente di determinazione* $R^2$ (cioè, la proporzione nella varianza dei risultati predicibile a partire dai valori di ingresso) di un approssimatore. Tale analisi è stata effettuata prima su 5 classi di utenti categorizzate con k-means sulla base dei propri consumi medi, poi, per meglio preservare le caratteristiche dei singoli utenti, sulle singole abitazioni.

I risultati della feature selection sulle singole abitazioni sono esposti nella Tabella 5.1, quelli della feature selection sulle categorie nella Figura 5.3. In entrambi i casi,

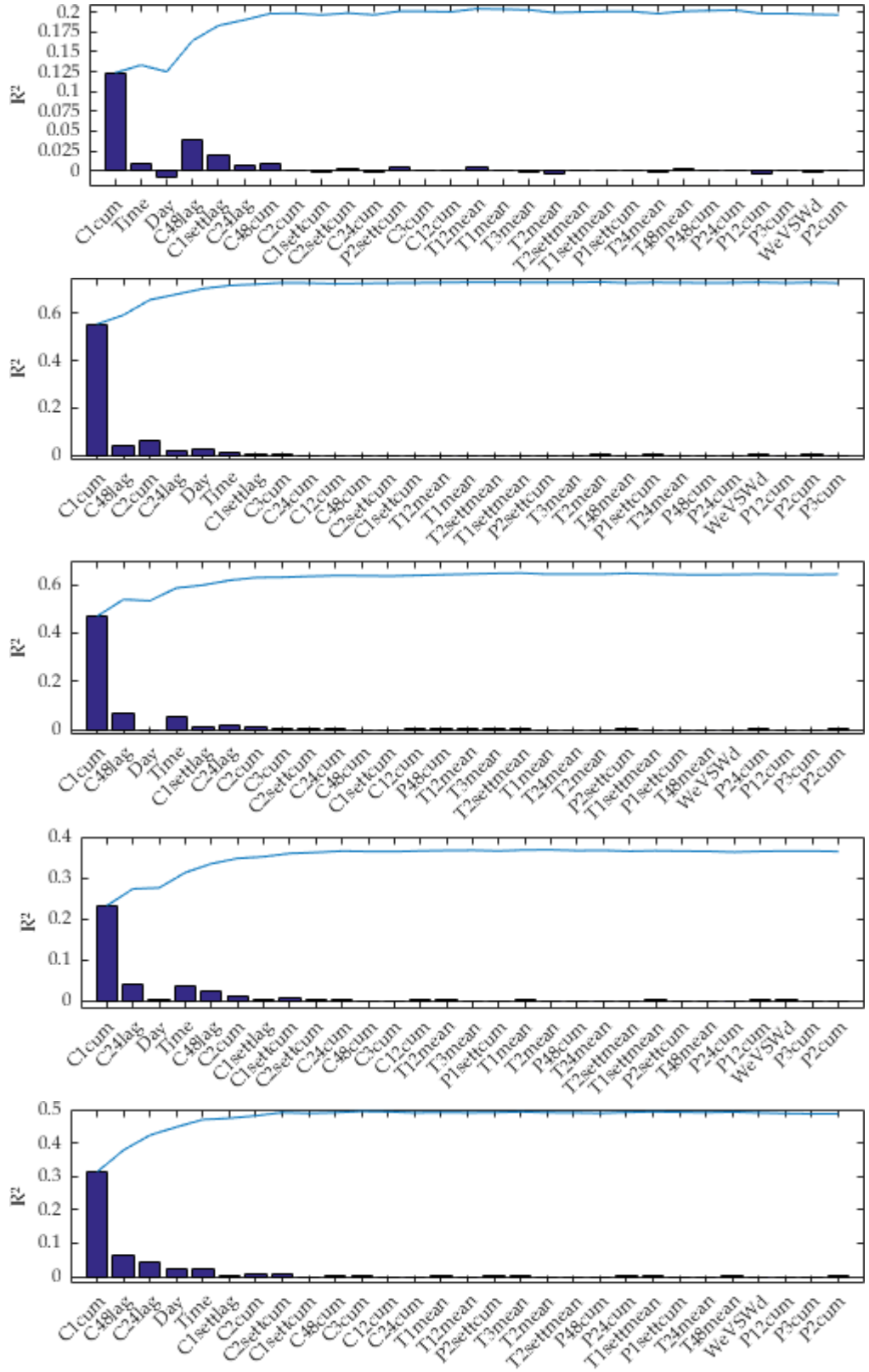



Figura 5.3: Risultati della feature selection sulle categorie di utenti. Si può notare la grande importanza della feature *C1cum*, e lo scarso contributo dato da *Day* e *Time*.

le feature selezionate più spesso tendono ad essere quelle relative a consumi puntuali, sia vicini che lontani nel tempo, così come le feature temporali (giorno e ora); in particolare il consumo dell'ora precedente a quella attuale è la feature più importante in assoluto al fine di una buona predizione. Questo pare confermare alcune delle nostre supposizioni preliminari.

Al contrario, non pare esistere una dipendenza significativa dalle feature meteorologiche: esse sono selezionate raramente, e nella maggioranza dei casi in posizione bassa. Tale scarsa influenza potrebbe essere dovuta al fatto che, dato che i dati a nostra disposizione si riferiscono ad un periodo inferiore ad un anno, non è possibile estrarre da essi informazioni significative a proposito delle tendenze stagionali.

Come ulteriore nota, facciamo notare che le feature temporali, per quanto selezionate in posizione alta da molte abitazioni, tendono a fornire un contributo basso o addirittura negativo alla qualità di $R^2$, forse a causa del modo da noi scelto per rappresentarle (cioè, come semplici numeri interi compresi in 0–23 e 1–7 per ora e giorno rispettivamente). $R^2$ presenta comunque notevoli variazioni tra un'abitazione e l'altra, e in rari casi raggiunge valori molto elevati.

Abbiamo quindi scelto di utilizzare le 7 feature selezionate da più abitazioni:

- Ora del giorno (Time).
- Giorno della settimana (Day).
- Consumo che precede di un'ora quello attuale (C1cum).
- Consumo che precede di due ore quello attuale (C2lag).[6]
- Consumo che precede di 24 ore quello attuale (C24lag).
- Consumo che precede di 48 ore quello attuale (C48lag).
- Consumo che precede di una settimana (168 ore) quello attuale (C1settlag).[7]

Ad esse è aggiunta una feature costante a 1 che funge da intercetta, catturando quanto non già descritto da queste feature.

Le funzioni base sono semplici funzioni identità che replicano uno specifico valore dello stato senza variazioni:

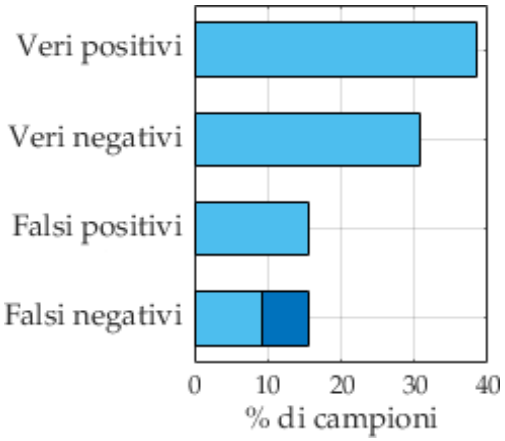
$$\phi_i(\boldsymbol{s}) = s_i.$$

Abbiamo inoltre constatato che normalizzare le feature Time e Day dividendole per il loro valore massimo (quindi, Time è divisa per 23 e Day per 7) porta a risultati migliori nell'approssimatore finale e riduce i tempi di calcolo. Tale normalizzazione è effettuata non nelle funzioni base ma direttamente nei dati d'ingresso.

In definitiva, la politica è caratterizzata da 24 parametri, cioè le 8 features moltiplicate per 3 i parametri della distribuzione congiunta.

[6] Tecnicamente, questa esatta feature non è stata considerata nelle nostre analisi. Tuttavia, data la presenza di C1cum essa fornisce le stesse informazioni di C2cum (dato che, per definizione, $C2cum = C1cum + C2lag$).

[7] Data la necessità di avere a disposizione almeno 1 settimana di dati per poter usare questa feature, il numero di traiettorie disponibili per ogni abitazione si riduce a $5352 - 24 \times 7 = 5184$.




| | Positivi | Negativi | Totale |
|---|---|---|---|
| Veri | 38,38% | 30,67% | 69,05% |
| Falsi | 15,47% | 15,48% | 30,95% |
| Totale | 53,85% | 46,15% | |

Figura 5.4: Matrice di confusione media della parte bernoulliana della politica, espressa in forma grafica e tabulare. L'area più scura nel grafico indica i falsi negativi in casi in cui l'azione reale è non superiore a 10 litri, soglia scelta per rappresentare piccoli eventi di consumo quale ad esempio un singolo sciacquone.

### 5.3.3 Risultati

La politica da noi definita si è dimostrata in grado di approssimare con un sufficiente grado di precisione la maggioranza delle abitazioni da noi considerate: non abbiamo quindi ritenuto opportuno implementare classi di politiche personalizzate. Le stime dei parametri delle due parti (bernoulliana e lognormale) sono state effettuate separatamente e usando metodi diversi, spiegati nelle prossime sezioni.

#### Parte bernoulliana

**Comportamento** I pesi della parte bernoulliana sono stati ottenuti tramite regressione logistica[8] sui dati. Il suo comportamento medio è mostrato in Figura 5.4, tali valori sono stati ottenuti:

- calcolando i pesi delle singole abitazioni
- per ogni abitazione, facendo generare alla politica di esempio 5 esperimenti
- per ogni esperimento, calcolando il numero di coincidenze con il dataset reale
- mediando i risultati dei 5 esperimenti per ogni abitazione
- mediando i risultati complessivi.

Come si può notare, in media l'approssimatore dà un risultato corretto in circa il 69% dei casi; inoltre, in media circa il 41% dei falsi negativi si verifica in casi in cui il consumo reale è contenuto (entro 10 litri). La percentuale complessiva di successo non è mai inferiore al 52% circa, e nei casi migliori (principalmente per le abitazioni con un numero estremamente basso o estremamente alto di consumi in totale, come prevedibile) può anche superare il 99%.

[8]Tale stima è stata effettuata usando la funzione di regressione logistica fornita dal pacchetto mlpack (`http://www.mlpack.org/`).

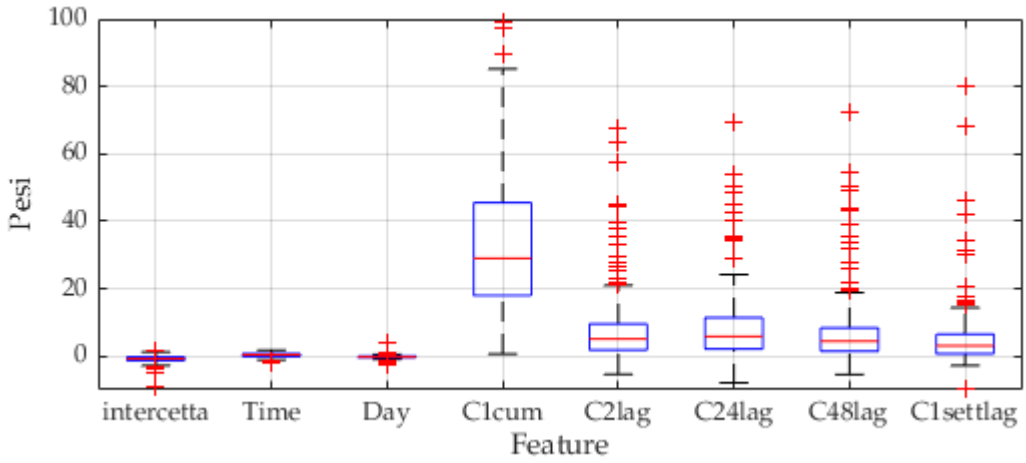



Figura 5.5: Distribuzione dei pesi della parte bernoulliana della politica, considerando tutte le abitazioni ed escludendo gli outlier più estremi.

È inoltre interessante notare che non vi è una tendenza specifica a stimare in positivo o in negativo: in tutti i casi, infatti, le percentuali di falsi positivi e falsi negativi sono sempre molto simili tra loro.

**Caratterizzazione dei pesi** I pesi estratti sono visualizzati in Figura 5.5, omettendo gli outlier più estremi. Data la forma logistica del parametro $p(\boldsymbol{s})$ della distribuzione (§ 5.3.1) e considerando che le feature sono tutte non negative, in generale un peso positivo contribuisce ad aumentare la probabilità di consumare, e un peso negativo a diminuirla; tuttavia, il fatto che il valore finale di $p$ dipende anche dallo stato non permette di considerarla una corrispondenza esatta.

I pesi ottenuti sono coerenti con i risultati della feature selection. In particolare, il peso della feature *C1cum* è sempre positivo, ed è in generale il più elevato; le altre feature relative ai consumi pregressi hanno un comportamento simile. Al contrario, *Time* e *Day* hanno pesi molto piccoli.

In generale, abbiamo riscontrato una tendenza ad avere un peso di *C1cum* particolarmente elevato in abitazioni con consumi continui nel tempo: ciò è coerente con la forma logistica del parametro $p$, significa infatti che se tali abitazioni hanno appena consumato avranno un forte incentivo a continuare a consumare. Le abitazioni con numero globale di consumi più basso in assoluto, invece, hanno molti pesi fortemente negativi (non mostrati nel grafico): anche questo è sensato, dato che i pesi negativi in generale scoraggiano i consumi.

### Parte lognormale

Per la stima dei pesi abbiamo utilizzato un approssimatore biconvesso a massima verosimiglianza, cioè un approssimatore che, a partire da parametri iniziali arbitrari (nel nostro caso abbiamo impiegato pesi costanti a 1), raffina alternatamente le stime dei vettori dei pesi di media e precisione iterando fino a convergenza, puntando ad ogni passo a massimizzare la verosimiglianza delle osservazioni:

$$\max_{\boldsymbol{v}} \sum_i ln(f(\boldsymbol{s}_i, a_i)) \quad \Rightarrow \quad \frac{\partial}{\partial \boldsymbol{v}} \sum_i ln(f(\boldsymbol{s}_i, a_i)) = 0,$$

dove $v = \mu, \alpha$ e $f$ è la funzione di densità di probabilità della distribuzione.

Siamo riusciti a calcolare tale cifra di merito in forma chiusa per il vettore dei pesi della media, ma non per quello dei pesi della precisione a causa di questioni numeriche: abbiamo quindi sfruttato un semplice metodo di discesa gradiente per stimare il vettore di pesi della precisione.

**Comportamento** Nonostante la distribuzione non esattamente lognormale dei consumi reali, la nostra scelta di una distribuzione lognormale si è dimostrata in generale tale da fornire una stima adeguata dei profili di consumo.

Alcuni risultati di esempio sono mostrati in Figura 5.6. Tali grafici mostrano i consumi reali di 4 abitazioni eterogenee confrontati in ciascun caso con quelli di 3 esperimenti generati dalle politiche che usano i pesi da noi estratti, limitatamente ai casi di consumo reale positivo (cioè, quelli in cui si attiva la parte lognormale della politica). A causa della sua natura stocastica, è difficile che che la politica stimata generi esattamente gli stessi valori della politica reale; l'andamento macroscopico dei consumi reali è comunque in gran parte preservato.

Tuttavia, abbiamo riscontrato una marcata tendenza a sovrastimare in modo anche estremo i picchi di consumo: questo potrebbe essere dovuto al fatto che tali picchi, proprio perché eccezionali, esulano dal comportamento più regolare che la classe di politica da noi scelta può rappresentare; o forse, almeno in parte, alla mancanza di informazioni chiave o al metodo di stima non esatto che abbiamo utilizzato per ottenere il vettore $\alpha$.

**Caratterizzazione dei pesi** I pesi sono visualizzati in Figura 5.7, come sopra omettendo gli outlier più estremi. Alcuni di tali outlier sono, sia nel caso della media che in quello della precisione, i pesi di alcune delle abitazioni quasi totalmente prive di consumi non nulli, che a volte assumono valore in modulo molto elevato: ciò potrebbe essere dovuto a difficoltà di convergenza a causa dello scarso numero di campioni disponibili. Inoltre, tali abitazioni presentano spesso diversi pesi nulli nel vettore della media: data la scarsità di consumi, è plausibile che alcune feature legate ai consumi passati non influiscano in alcun modo sui parametri; a tali pesi nulli corrisponde sempre un peso uguale a 1 nel vettore dei pesi della precisione.

I pesi della media sono mostrati in Figura 5.7a: quelli relativi ai consumi pregressi sono quasi sempre positivi, e ancora una volta tendono ad essere i più rilevanti. L'eccezione è il peso di *C2lag*, che tende ad essere equamente spartito tra positivo e negativo a seconda dei casi. Il peso dell'intercetta, al contrario, è sempre negativo. I pesi delle feature temporali sono ancora bassi, coerentemente con quanto evidenziato in fase di feature selection.

L'andamento medio dei pesi della precisione, mostrati in Figura 5.7b, è approssimativamente speculare a quello dei pesi della media; tuttavia i pesi della precisione presentano un numero maggiore di outlier sia in positivo che in negativo, soprattutto per quanto riguarda i pesi delle feature relative ai consumi lontani. Sono inoltre distribuiti in un intervallo notevolmente più ristretto. In generale, i pesi della precisione presentano più instabilità: ciò potrebbe avere essere in parte dovuto al metodo non esatto con cui sono stimati.

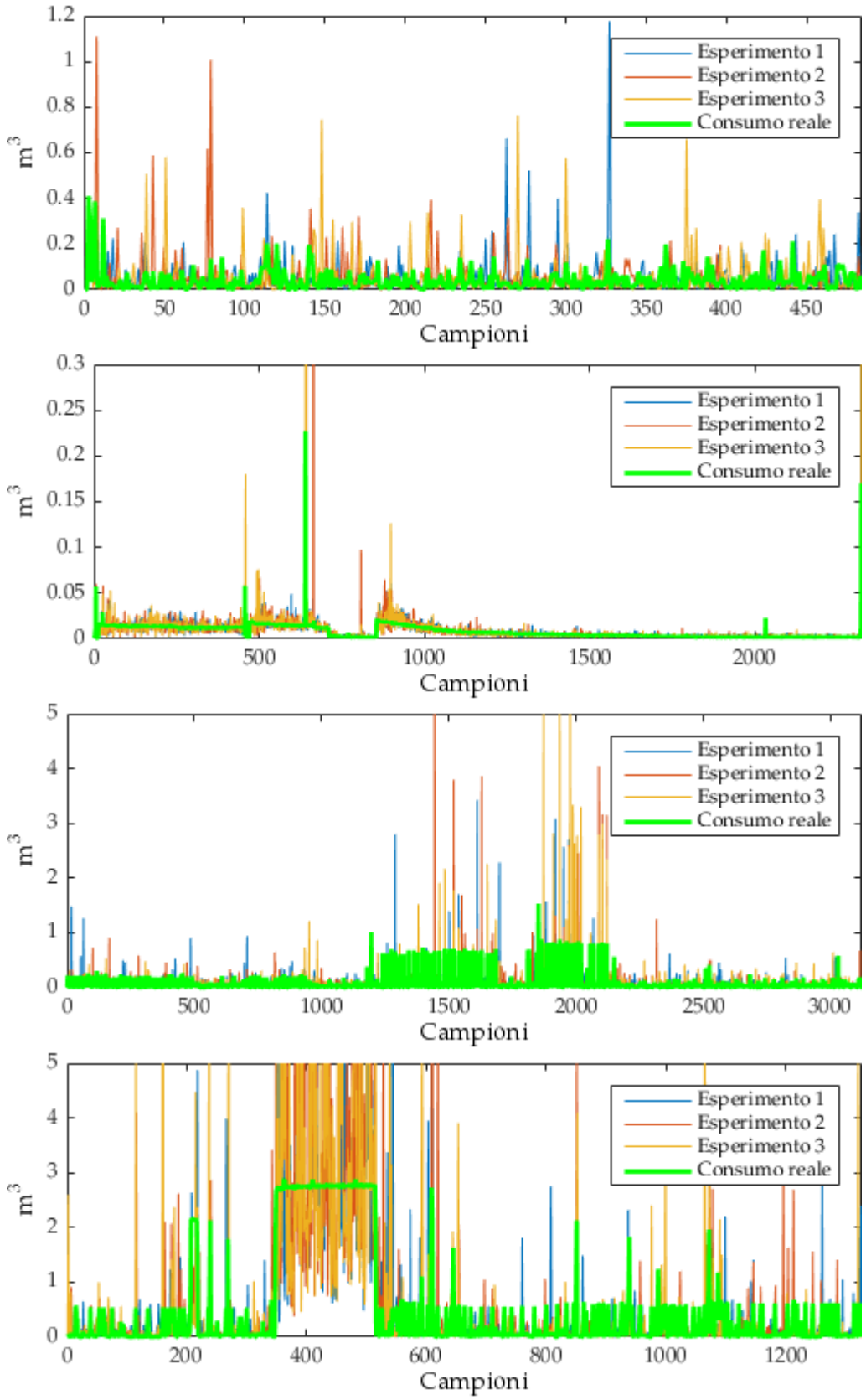
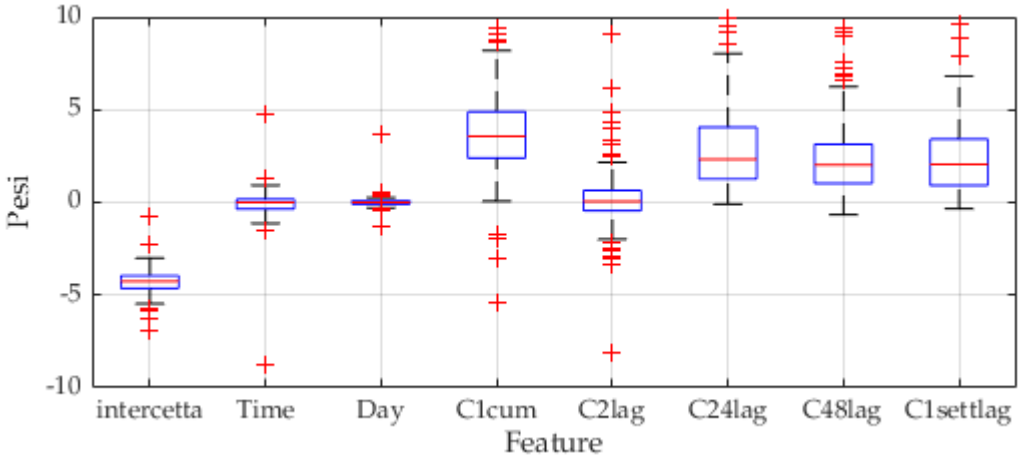
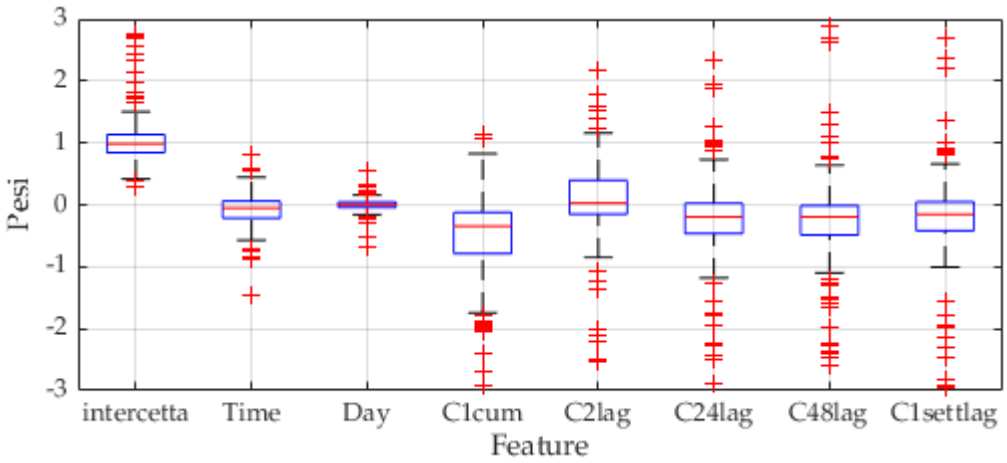



Figura 5.6: Alcuni esempi di risultato della parte lognormale della politica. Si può notare l'andamento macroscopico affine a quello reale, così come la sovrastima dei picchi.




(a) Pesi della media.




(b) Pesi della precisione.

Figura 5.7: Distribuzione dei pesi della parte lognormale della politica, considerando tutte le abitazioni ed escludendo gli outlier più estremi.

## 5.4 Definizione della funzione premio

### 5.4.1 Scelta della classe di funzione premio

Abbiamo ritenuto sufficiente la comune assunzione di una semplice parametrizzazione lineare (eq. 2.7). Questo, eventualmente, permette di sfruttare PGIRL (§ 2.3.4).

### 5.4.2 Scelta delle feature della funzione premio

Nel corso delle nostre analisi, abbiamo considerato diverse categorie di feature:

- una funzione che rappresenta il costo dell'acqua e che funge da disincentivo generico per il consumo, implementata semplicemente come $\phi_i(\boldsymbol{s}, a) = -a$.
- funzioni che rappresentano il trovarsi o meno in determinate fasce orarie, o in determinati giorni della settimana.
- funzioni che combinano in vari modi il consumo corrente con i consumi passati.

Prendendo come spunto l'analisi descritta in § 3.2.2, abbiamo deciso di utilizzare prevalentemente feature volte a rappresentare l'abitudinarietà degli utenti, per avere in tale studio un termine di paragone.

Seguendo l'esempio delle feature della politica, abbiamo scelto come componenti i conteggi di consumi in tre intervalli temporali, ciascuno normalizzato dividendo per il proprio massimo possibile al fine di ottenere valori compresi in $[0, 1]$:

- consumi di 1-2 ore prima di quella corrente, volti a rappresentare i consumi temporalmente locali.
- consumi di 24, 48, … ore prima di quella corrente, per un periodo di due settimane (quindi fino a 336 ore), volti a rappresentare la routine giornaliera recente.
- consumi di 1 e 2 settimane (168 e 336 ore) prima dell'ora corrente, volti a rappresentare la routine settimanale recente.

Abbiamo scelto questi intervalli limitati sia per modellizzare il fatto che il nostro sistema di riferimento – cioè l'utente – si può supporre con memoria limitata, sia per evitare di scartare troppi dati: infatti, avendo bisogno di 2 intere settimane di dati non è possibile sfruttare come stati i campioni delle prime 2 settimane; questo ci porta a considerare un'ulteriore settimana iniziale in meno rispetto a quanto fatto durante il calcolo dei pesi della politica.

Abbiamo poi suddiviso i consumi orari nelle stesse 4 classi usate in Cominola et al., 2016, cioè:

- *c0*: consumi nulli (0 litri/ora).
- *c1*: consumi bassi (1–12 litri/ora).
- *c2*: consumi medi (13–100 litri/ora).
- *c3*: consumi alti (più di 100 litri/ora).

Il gruppo finale di feature è stato ottenuto incrociando tra loro questi due insiemi: abbiamo così ottenuto 12 feature, ognuna delle quali rappresenta il conteggio dei consumi appartenenti a ciascuna classe nell'intervallo temporale considerato (ad esempio, quante volte su 14 si è verificato un consumo basso in questa stessa ora in tutti i giorni delle ultime 2 settimane). Tali valori sono in ogni stato moltiplicati per l'azione corrente, per avere feature effettivamente dipendenti sia dallo stato che dall'azione.

Ad esse abbiamo aggiunto la funzione costo definita in precedenza e una funzione binaria che rappresenta il trovarsi o meno in un giorno lavorativo (lunedì–venerdì), anch'essa moltiplicata per il consumo corrente.

La funzione premio da noi definita comprende quindi 14 feature:

- la funzione costo (nei grafici indicata con *costo*).
- 12 funzioni che rappresentano conteggi normalizzati di consumi appartenenti ad una certa fascia in un certo periodo e moltiplicati per il consumo corrente (indicate con $\{2h = 1$ e 2 ore prima; $d = 24, 48, \ldots, 24\times14$ ore prima; $w = 24\times7$ e $24\times14$ ore prima$\} \times \{c0; c1; c2; c3\}$).
- la funzione che rappresenta il trovarsi o meno in un giorno lavorativo moltiplicata per il consumo corrente ($wd \times c$).

### 5.4.3 Risultati

Abbiamo constatato che, se si utilizzano combinazioni di feature che ricoprono senza sovrapposizioni l'intero spazio dei possibili valori di una categoria, durante la sua fase di pre-processing (§ 2.3.4) PGIRL rimuove spesso una o più feature, in quanto considerate linearmente dipendenti tra loro: ciò è corretto dato che, ad esempio, tutti i consumi verificatisi in un certo intervallo appartengono per forza ad una delle 4 classi, quindi se si sa quanti consumi appartengono a 3 delle 4 classi il numero di consumi appartenenti alla quarta è univocamente determinato. Questa rimozione, pur non essendo intrinsecamente errata, ha il difetto di rendere più difficile l'interpretazione dei risultati: di conseguenza, abbiamo scelto di utilizzare la versione base di GIRL.

La distribuzione dei pesi dei vari utenti è mostrata in Figura 5.8. Si può notare che i pesi delle feature rappresentanti la routine (cioè tutte a parte prima e ultima) tendono a seguire un andamento mediamente simile a prescindere dall'intervallo temporale considerato.

Abbiamo constatato che GIRL tende a priori ad assegnare all'ultima feature pesi la cui distribuzione si estende verso il basso: ciò avviene anche in questo caso; tuttavia dalla fase di feature selection (§ 5.3.2) è noto che le feature dipendenti dal giorno della settimana hanno importanza bassa, quindi è legittimo che alla feature dipendente dal giorno lavorativo siano assegnati pesi globalmente bassi.

## 5.5 Clustering

Abbiamo scelto un algoritmo di tipo EM (§ 2.4.1) basato su *mistura di gaussiane*, cioè che suppone che i dati siano approssimabili da un certo numero di distribuzioni gaussiane multivariate (McLachlan e Peel, 2000). Per semplicità, tale clustering è netto, cioè ogni elemento può essere assegnato ad un unico cluster.

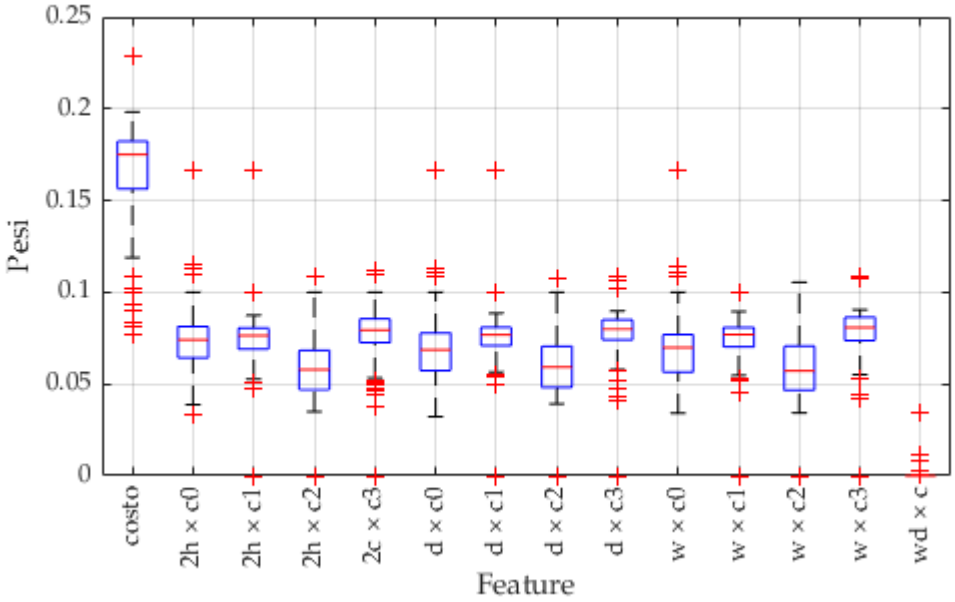



Figura 5.8: Distribuzione dei pesi della funzione premio, considerando tutte le abitazioni. Nell'ordine: costo dell'acqua; feature incrociate intervallo di tempo/classe di consumo; giorno feriale. Tutte le feature a parte la prima sono internamente moltiplicate per il consumo attuale.

Per fare ciò, ci siamo avvalsi delle funzioni fitgmdist[9] e cluster[10] fornite da MATLAB®: la prima, data una matrice di dati e un parametro K, estrae un insieme di K gaussiane multivariate adatto a rappresentare i dati, la seconda effettua il clustering EM vero e proprio sfruttando tali gaussiane.

Abbiamo constatato che se fitgmdist è applicata ai dati completi tende a fallire, potenzialmente a causa di dipendenze lineari tra i pesi: il clustering è quindi effettuato sui pesi elaborati dall'algoritmo *t-SNE* (van der Maaten e Hinton, 2008) per ridurne la dimensionalità da 14 a 2.

Il numero di cluster da usare è stato scelto considerando prove empiriche e un'analisi di silhouette (Kaufman e Rousseeuw, 1990): diversi esperimenti hanno dimostrato che 3–5 cluster portano a risultati migliori.

### 5.5.1 Risultati

La suddivisione definitiva è stata ottenuta effettuando diverse prove (si ricorda che EM è un algoritmo non deterministico) con 4 cluster: questo ha portato a 3 cluster di dimensioni comparabili tra loro accompagnati da un quarto notevolmente più piccolo.

Per ogni cluster, la Figura 5.9 mostra la distribuzione dei pesi delle funzioni premio delle abitazioni in esso contenute, mentre la Figura 5.10 mostra i profili orari complessivi.

Si può notare una certa correlazione tra entità media dei consumi e peso dato alla funzione costo: ad esempio, il cluster 2, per cui i pesi della funzione costo tendono a

[9] `http://www.mathworks.com/help/stats/fitgmdist.html`
[10] `http://www.mathworks.com/help/stats/gmdistribution.cluster.html`

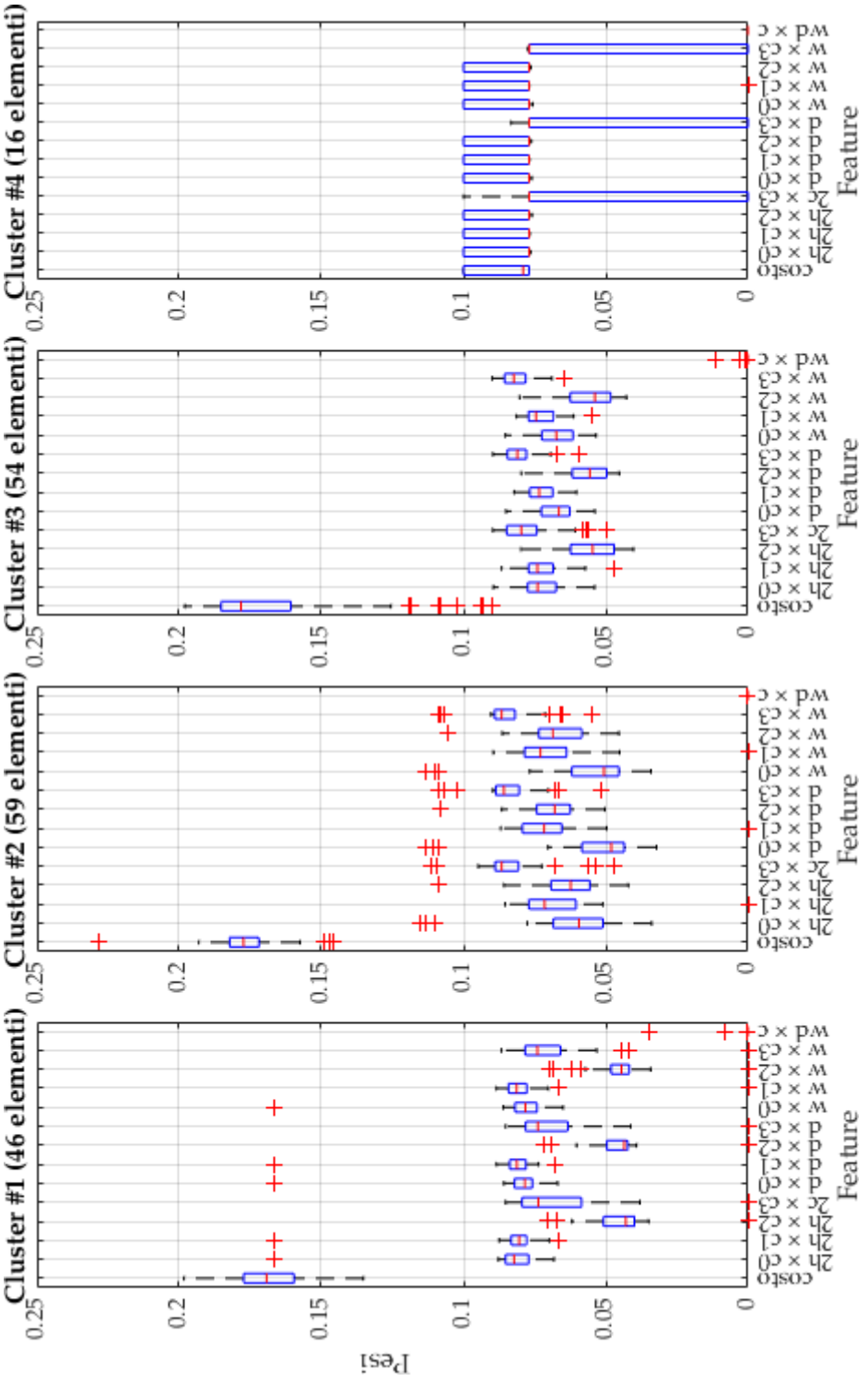



Figura 5.9: Distribuzione dei pesi della funzione premio distinti per cluster.

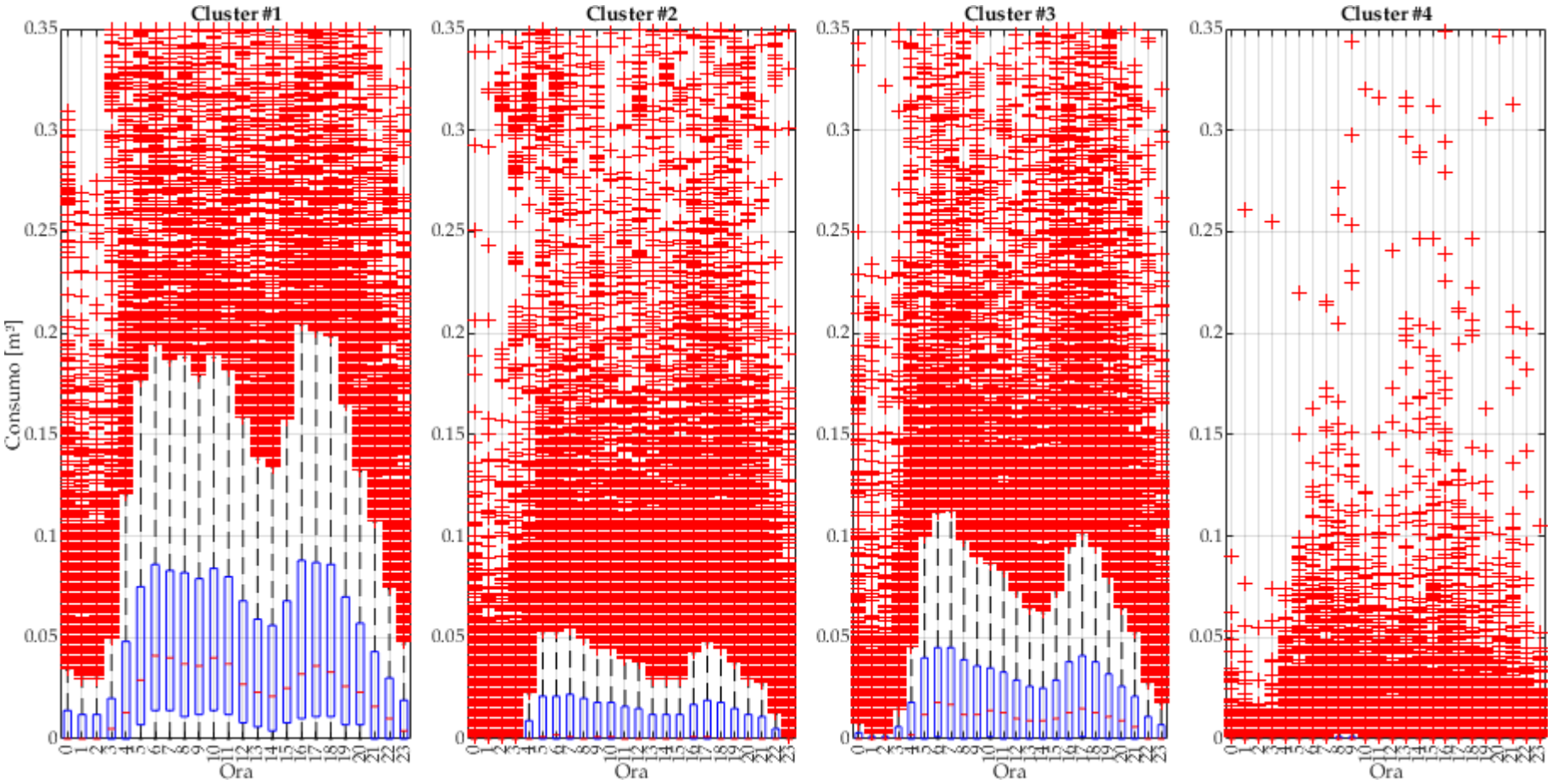



Figura 5.10: Profili orari delle abitazioni appartenenti ad ogni cluster. Ad ogni etichetta oraria corrispondono i consumi cumulativi dell'ora precedente. Per semplicità nei calcoli, le ultime due settimane in ora solare non sono state considerate.

concentrarsi più in alto rispetto ai cluster 1 e 3, è quello con i consumi più bassi tra i 3. Altri nostri esperimenti tendono a mostrare una suddivisione ancor più significativa, soprattutto usando k-means (§ 2.4.1) come algoritmo di clustering; tuttavia, dato che tali esperimenti non hanno portato ad altri risultati degni di nota non riteniamo opportuno approfondirli.

I pesi del cluster 4 sono spesso uguali o molto simili tra loro. Ciò è dovuto al fatto che tale cluster è prevalentemente composto da abitazioni con una quantità estremamente bassa – in alcuni casi anche inferiore all'1% – di consumi non nulli: di conseguenza, GIRL non ha a disposizione informazioni sufficienti ad una stima accurata dei parametri. È comunque degno di nota il fatto che il clustering sia stato in grado di isolare tale gruppo di abitazioni insolite.

La Figura 5.11 mostra il primo eigenbehavior di ciascuna abitazione, raggruppando le abitazioni secondo i cluster trovati. Si può notare che, pur con qualche eccezione, gli eigenbehavior delle abitazioni appartenenti ad uno stesso gruppo sono simili, e ben distinguibili da quelli degli altri gruppi.

L'efficacia del nostro approccio è constatabile confrontanto le figure 5.10 e 5.11. In particolare, si può notare che:

- Il cluster 1, che dimostra una preferenza per consumi medi e alti, è effettivamente quello con i consumi globalmente più elevati.
- Gli eigenbehavior del cluster 3 sono simili a quelli del cluster 1, ma con una maggiore preferenza per i consumi nulli rispetto a quelli medi e alti: i profili globali di consumo dei due clusters sono simili, ma quello di 3 è mediamente più basso.
- Il cluster 2 ha una preferenza marcata per i consumi nulli, e una lieve preferenza per i consumi bassi e medi: tra i primi 3 clusters, 2 è quello con il profilo globale di consumo più basso.
- Il cluster 4 ha una preferenza estremamente marcata per i consumi nulli, e dà poca o nessuna importanza alle altre classi: ciò è coerente con il fatto che le abitazioni di tale cluster hanno spesso un numero estremamente basso di consumi in totale, e di conseguenza questo cluster ha il profilo di consumo globale più basso di tutti.

Abbiamo però notato che questa interpretazione sembra non concordare con i pesi trovati: ad esempio, ci saremmo aspettati che nel caso del cluster 1 i pesi delle features collegate ai consumi medi fossero i più alti, dato che per tale cluster la classe 2 è prevalente. Invece, i pesi associati alle feature relative ai conteggi di consumi tendono a comportarsi in maniera opposta.

Per individuare la natura di tale discrepanza, abbiamo quindi calcolato, per tutte le abitazioni, le feature expectation (eq. 2.8) delle singole features, raccogliendole poi negli stessi cluster individuati sopra. I risultati complessivi sono esposti nella Figura 5.12. In particolare, si può notare che:

- la distribuzione della feature expectation relativa ai consumi in giorni feriali è circa l'inversa di quella relativa al costo dell'acqua, ma lievemente inferiore in modulo: ciò è legittimo, dato che tali funzioni sono essenzialmente l'una l'inversa dell'altra ma la prima si attiva in un numero inferiore di giorni.

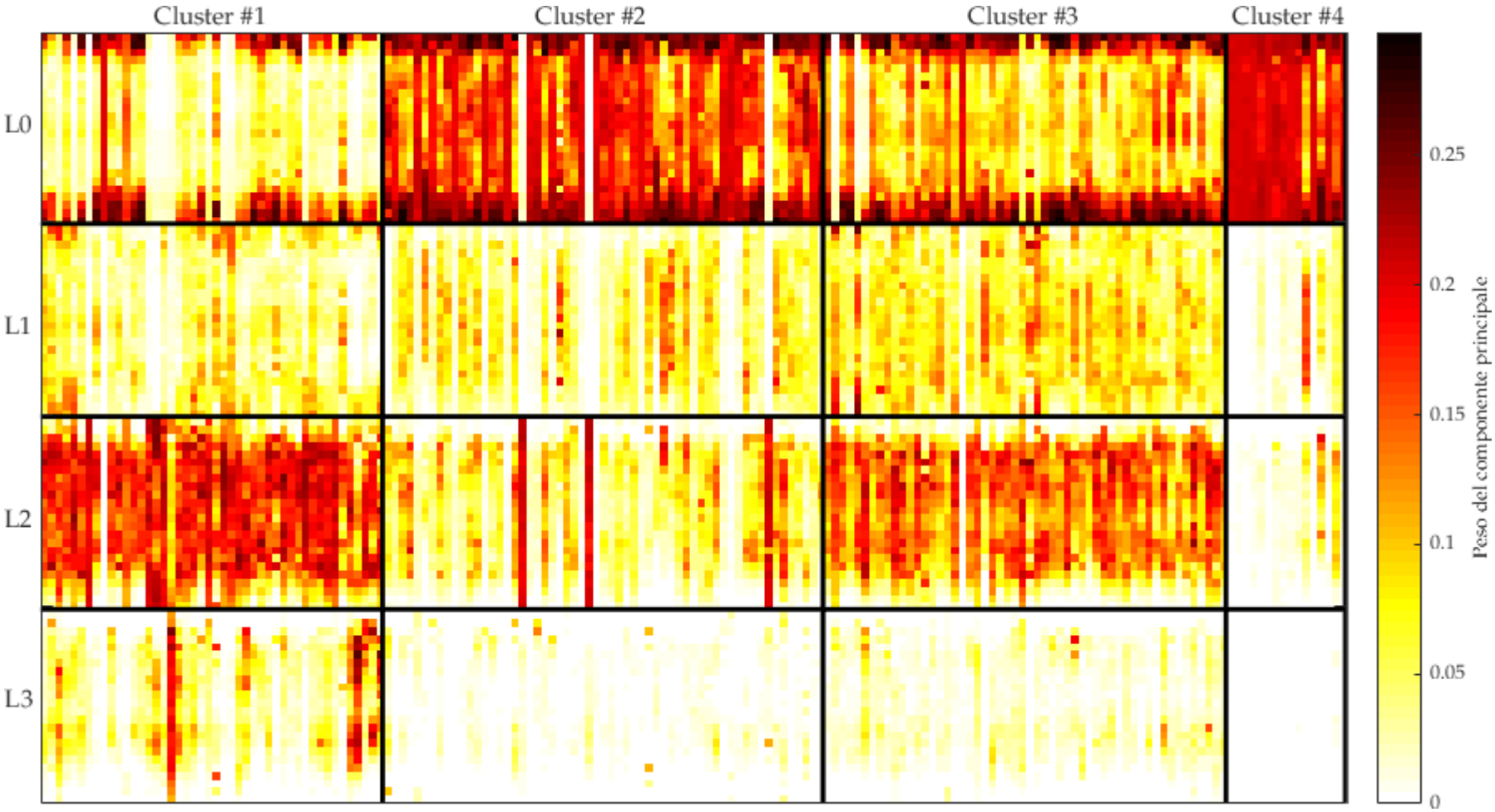



Figura 5.11: Confronto tra le categorie di utenti basato sul primo eigenbehavior. Il significato di ciascun elemento è lo stesso della Figura 3.1.

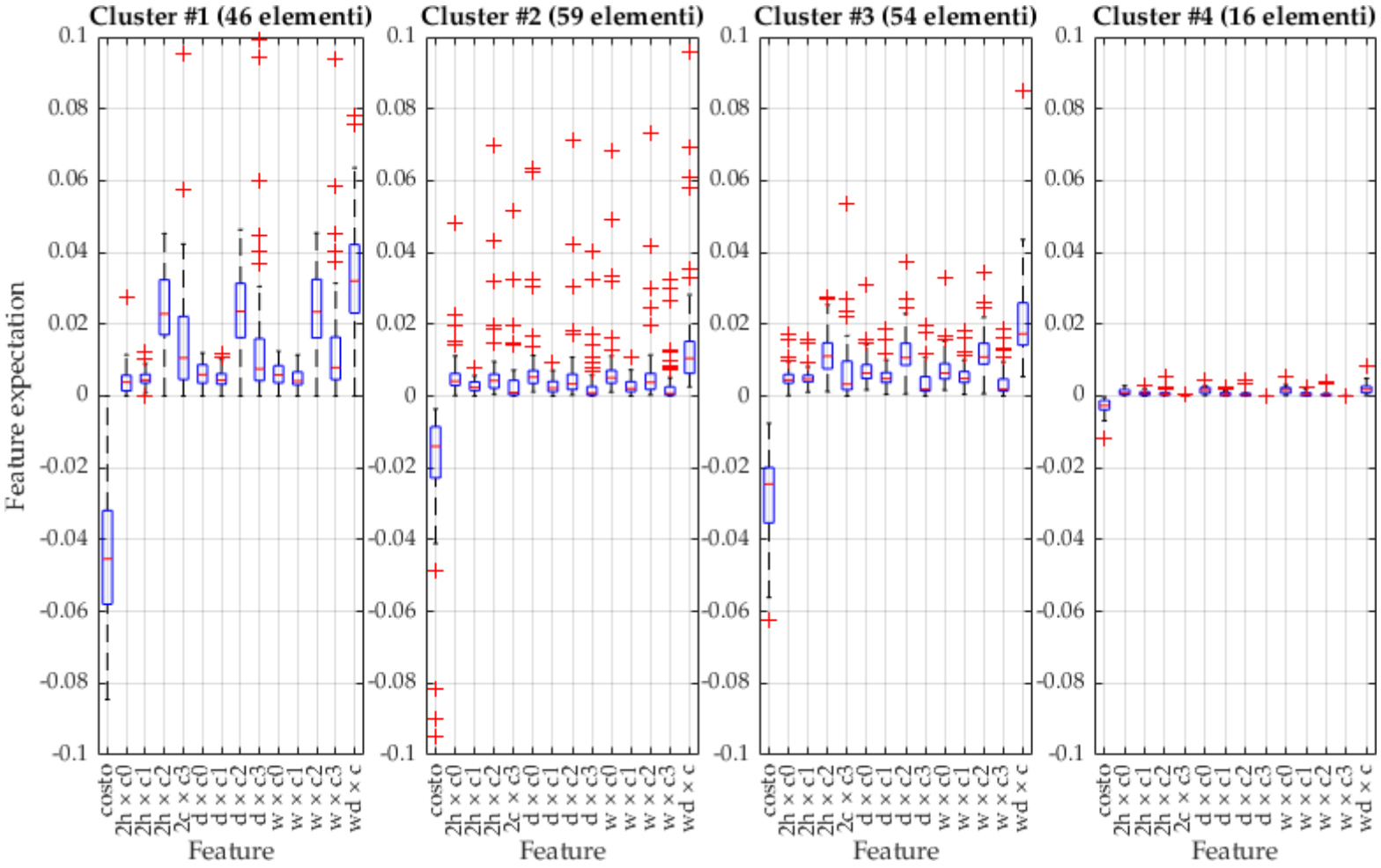



Figura 5.12: Distribuzione delle feature expectation per ogni cluster.

- soprattutto, i valori delle feature expectation delle feature intervallo di tempo $\times$ classe di consumo sono in media l'inverso dei pesi corrispondenti della funzione premio.

Inoltre, nella Figura 5.13 è mostrato, ancora per ognuno dei nostri cluster, il prodotto di ciascun peso per la relativa feature expectation (la funzione costo, nel cui caso tale prodotto è globalmente compreso in un intervallo di circa -0,025 – 0, è omessa per rendere le altre più visibili). Si può notare che l'influenza della feature expectation è dominante rispetto a quella dei pesi; l'apparente discrepanza da noi individuata è quindi da imputarsi alla differenza di scala tra le features.

Analizzando i grafici in Figura 5.9 e interpretando, in luce del confronto con la feature expectation, i pesi bassi come dominanti, si può notare che:

- il cluster 1 dà grande importanza ai consumi medi. L'importanza data ai consumi alti è in generale più alta rispetto agli altri clusters, ma presenta comunque forte variabilità. Anche i consumi nulli hanno una discreta importanza, soprattutto per quanto riguarda i consumi giornalieri e settimanali: dato il grafico in Figura 5.11 si può ipotizzare che ciò sia dovuto al riconoscimento della tendenza a non consumare nelle ore notturne.

- il cluster 2 dà importanza elevata ai consumi nulli e scarsa ai consumi alti, coerentemente con l'analisi degli eigenbehavior.

- il cluster 3 dà una certa importanza ai consumi nulli, soprattutto nel caso delle features relative a consumi giornalieri e settimanali: come nel caso del cluster 1, questo potrebbe essere dovuto ai consumi notturni nulli. Presenta anche una forte influenza dei consumi medi, anche se in generale inferiore a quella del cluster 1. Entrambi questi risultati sono coerenti con l'analisi degli eigenbehavior.

- i pesi del cluster 4, invece, non sembrano corrispondere a tendenze significative: ciò è legittimo, dato che la loro peculiare distribuzione è segno di problemi durante l'esecuzione dell'algoritmo causati dalla struttura dei dati ad esse corrispondenti.

L'influenza delle feature relative ai consumi nulli appare lievemente inferiore a quanto il confronto con eigenbehavior farebbe supporre: questo potrebbe essere dovuto alla formulazione semplificata da noi impiegata, in cui un'azione nulla azzera le feature in cui è coinvolta (tutte, nel nostro caso). Inoltre, come nel caso globale mostrato in Figura 5.8 anche all'interno dei singoli cluster la fascia di consumo ha un'influenza maggiore sul peso rispetto all'intervallo di tempo considerato.

In definitiva, l'approccio da noi scelto è stato in grado di effettuare una categorizzazione significativa degli utenti. I risultati permettono di riconoscere le abitudini di gruppi di utenti, e forniscono indicazioni interessanti a proposito della variabilità temporale dei loro consumi, codificate nelle distinzioni tra feature relative a diversi intervalli temporali. Inoltre, le anomalie quali abitazioni con globalmente pochi consumi sono riconosciute e isolate.

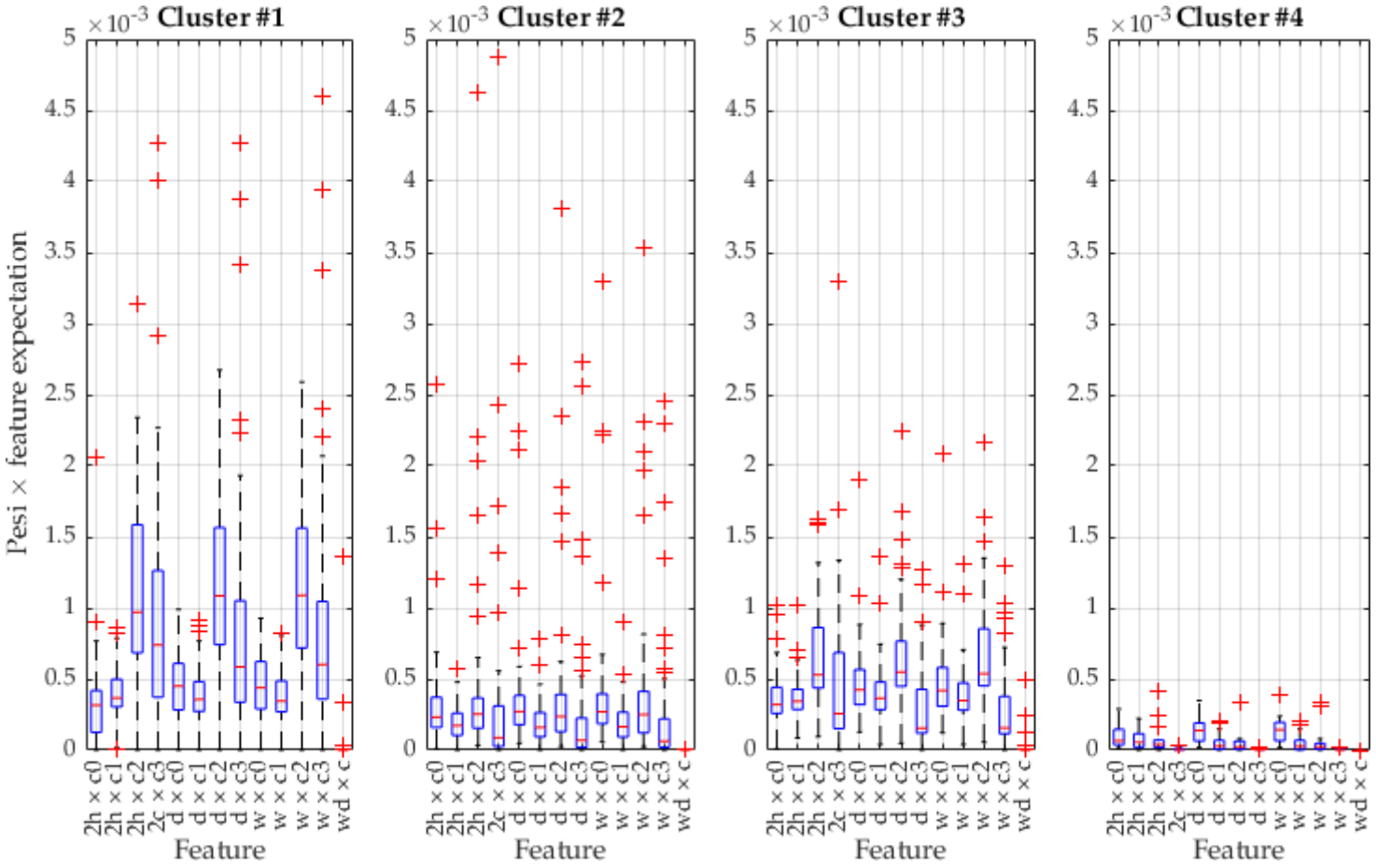



Figura 5.13: Prodotto tra pesi delle features della funzione premio e relative feature expectation, funzione costo esclusa.

# 6 Conclusioni e sviluppi futuri

In questa tesi abbiamo proposto un semplice procedimento che combina un algoritmo IRL – nello specifico GIRL – con tecniche di clustering applicate a posteriori: il nostro scopo è individuare una rappresentazione delle motivazioni degli utenti di un sistema, in particolare gli utenti domestici di una rete idrica, sotto forma di funzioni premio, sulla base delle quali sia possibile raggruppare gli utenti in categorie sulla base di interessi comuni.

Nonostante la scarsità di dati a nostra disposizione, siamo stati in grado di ottenere un modello tale da portare ad una categorizzazione significativa degli utenti basata sulle loro abitudini, in particolare sulle loro preferenze verso una specifica fascia di consumo (consumi nulli, bassi, medi o alti). Abbiamo constatato una corrispondenza tra le informazioni fornite dalle funzioni premio ottenute dal nostro metodo e quelle ottenute da un altro metodi che punta in modo più diretto ad individuare regolarità comportamentali per mezzo di un'analisi basata su eigenbehavior. Inoltre, il nostro metodo è stato in grado di riconoscere ed isolare diverse abitazioni dal comportamento anomalo.

Tuttavia, l'abitudinarietà non è che uno dei possibili aspetti degli utenti che possono essere considerati: ulteriori studi potrebbero quindi concentrarsi su altre caratteristiche, come ad esempio la distribuzione oraria dei consumi.

In particolare, il fatto che nel nostro caso le feature esogene (temporali e soprattutto meteorologiche) abbiano dimostrato un'influenza scarsa non significa che ciò sia necessariamente vero in generale: dati provenienti da altre regioni, con diverse caratteristiche climatiche e sociali, potrebbero portare a risultati differenti. Sfruttare rilevamenti su un periodo più esteso, eventualmente anche di alcuni anni, potrebbe portare a riconoscere tendenze stagionali. Inoltre, potrebbe essere interessante applicare questo stesso approccio a dataset più ampi e ricchi, idealmente comprensivi di più informazioni quali ad esempio i dati psicografici a noi mancanti.

Abbiamo effettuato diverse assunzioni semplificative a proposito della rappresentazione dei dati. Scegliere una classe di politica – ed eventualmente di funzione premio – più ricca della nostra – ad esempio considerando una distribuzione più vicina alla reale struttura dei dati rispetto alla nostra bernoulliana/lognormale pura, eventualmente unita ad una parametrizzazione non lineare – potrebbe contribuire a risolvere alcuni dei problemi da noi incontrati quale ad esempio la sovrastima dei picchi. Inoltre, una rappresentazione sfocata (fuzzy) di fasce di consumo, fasce orarie e simili caratteristiche potrebbe essere più significativa della distinzione netta da noi usata.

Infine, ribadiamo che il nostro approccio è una semplice messa in serie di GIRL e clustering: prendendo spunto dagli algoritmi citati in § 2.4.2, potrebbe essere interessante definire un algoritmo che integri le due fasi in modo più stretto, con la proprietà aggiunta di essere in grado di considerare atomici gli agenti mantenendo raggruppate le loro traiettorie.

# Bibliografia


Abbeel, Pieter e Andrew Y. Ng (2004). «Apprenticeship Learning via Inverse Reinforcement Learning». In: *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. A cura di Carla E. Brodley. ACM (cit. alle pp. VII, 7, 8).

Babeş-Vroman, Monica, Vukosi N. Marivate, Kaushik Subramanian e Michael L. Littman (2011). «Apprenticeship Learning About Multiple Intentions». In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. A cura di Lise Getoor e Tobias Scheffer. Omnipress, pp. 897–904 (cit. alle pp. VIII, 1, 14).

Baxter, J. e P.L. Bartlett (2001). In: *Journal of Artificial Intelligence Research (JAIR)* 15, pp. 319–350 (cit. a p. 11).

Bellman, Richard Ernest (1957). *Dynamic Programming*. Princeton Univ. Press, New York (cit. a p. 5).

Bishop, Christopher (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York (cit. a p. 13).

Boularias, Abdeslam, Jens Kober e Jan Peters (2011). «Relative Entropy Inverse Reinforcement Learning». In: *Proceedings of the 14 International Conference on Artificial Intelligence and Statistics (AISTATS 2011)* (cit. a p. 9).

Brown, Martin e Robert Elliott Smith (2005). «Directed multi-objective optimization». In: *International Journal of Computers, Systems, and Signals* 6.1, pp. 3–17 (cit. a p. 11).

Busoniu, Lucian, Robert Babuska, Bart De Schutter e Damien Ernst (2010). *Reinforcement learning and dynamic programming using function approximators*. Vol. 39. CRC press (cit. a p. 5).

Choi, Jaedeug e Kee-Eung Kim (2012). «Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions». In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* A cura di Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou e Kilian Q. Weinberger, pp. 314–322 (cit. alle pp. VIII, 1, 15).

Cominola, A., M. Giuliani, D. Piga, Andrea Castelletti e A.E. Rizzoli (2015). «Benefits and challenges of using smart meters for advancing residential water demand modeling and management: A review». In: *Environmental Modelling & Software* 72, pp. 198–214. DOI: 10.1016/j.envsoft.2015.07.012 (cit. alle pp. VIII, 1, 17).

Cominola, Andrea, Andrea Moro, Luca Riva, Matteo Giuliani e Andrea Castelletti (2016). «Profiling residential water users' routines by eigenbehavior modelling». In: *Proceedings of the 8th International Congress on Environmental Modelling and Software*. Vol. 3. International Environmental Modelling e Software Society (iEMSs) (cit. alle pp. VIII, 2, 20, 21, 40).

Dempster, Arthur P., Nan M. Laird e Donald B. Rubin (1977). «Maximum likelihood from incomplete data via the EM algorithm». In: *Journal of the Royal Statistical Society, series B* 39.1, pp. 1–38 (cit. a p. 13).

Dimitrakakis, Christos e Constantin A. Rothkopf (2012). «Bayesian Multitask Inverse Reinforcement Learning». In: *Recent Advances in Reinforcement Learning*. A cura di Scott Sanner e Marcus Hutter. Vol. 7188. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 273–284. ISBN: 978-3-642-29945-2. DOI: 10.1007/978-3-642-29946-9_27 (cit. a p. 9).

Désidéri, Jean-Antoine (2012). «Multiple-gradient descent algorithm (mgda) for multiobjective optimization». In: *Comptes Rendus Mathematique* 350.56, pp. 313–318 (cit. a p. 12).

Eagle, Nathan e Alex Sandy Pentland (2009). «Eigenbehaviors: identifying structure in routine». In: *Behavioral Ecology and Sociobiology* 63.7, pp. 1057–1066. DOI: 10.1007/s00265-009-0739-0 (cit. alle pp. 2, 19).

Galelli, S. e A. Castelletti (2013). «Tree-based iterative input variable selection for hydrological modeling». In: *Water Resources Research* 49.7, pp. 4295–4310. ISSN: 1944-7973. DOI: 10.1002/wrcr.20339 (cit. a p. 32).

Hwang, Ching-Lai e Abu Syed Md. Masud (1979). *Multiple Objective Decision Making – Methods and Applications: A State-of-the-Art Survey*. Vol. 164. Springer Berlin Heidelberg (cit. a p. 11).

Inman, D. e P. Jeffrey (2006). «A review of residential water conservation tool performance and influences on implementation effectiveness». In: *Urban Water Journal* 3, pp. 127–143 (cit. a p. 18).

Kaufman, Leonard e Peter J. Rousseeuw (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc. (cit. a p. 42).

Klein, Edouard, Matthieu Geist, Bilal Piot e Olivier Pietquin (2012). «Inverse Reinforcement Learning through Structured Classification». In: *Advances in Neural Information Processing Systems 25*. A cura di F. Pereira, C.J.C. Burges, L. Bottou e K.Q. Weinberger. Curran Associates, Inc., pp. 1007–1015 (cit. a p. 8).

Klein, Edouard, Bilal Piot, Matthieu Geist e Olivier Pietquin (2013). «A Cascaded Supervised Learning Approach to Inverse Reinforcement Learning». In: *Machine Learning and Knowledge Discovery in Databases*. A cura di Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen e Filip Železný. Vol. 8188. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–16. ISBN: 978-3-642-40987-5. DOI: 10.1007/978-3-642-40988-2_1 (cit. a p. 8).

MacQueen, James B. (1967). «Some methods for classification and analysis of multivariate observations». In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, pp. 281–297 (cit. a p. 13).

McLachlan, Geoffrey e David Peel (2000). *Finite Mixture Models*. John Wiley & Sons, Inc. ISBN: 978-0-471-00626-8 (cit. a p. 41).

Moro, Andrea e Luca Riva (2016). «Inferring and clustering residential water consumers' routines by eigenbehavior modeling». Tesi di laurea mag. Politecnico di Milano, Italy (cit. a p. 17).

Ng, Andrew Y. e Stuart J. Russell (2000). «Algorithms for Inverse Reinforcement Learning». In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. A cura di Pat Langley. Morgan Kaufmann, pp. 663–670 (cit. alle pp. VII, 1, 7).

Pirotta, Matteo (2016). «Reinforcement learning: from theory to algorithms». Tesi di dott. Politecnico di Milano, Italy (cit. a p. 7).

Pirotta, Matteo e Marcello Restelli (2016). «Inverse Reinforcement Learning through Policy Gradient Minimization». In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)* (cit. alle pp. VII, 1, 10).

Ramachandran, Deepak e Eyal Amir (2007). «Bayesian Inverse Reinforcement Learning». In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence* 51, pp. 2586–2591 (cit. alle pp. 9, 15).

Ratliff, Nathan D., J. Andrew Bagnell e Martin A. Zinkevich (2006). «Maximum Margin Planning». In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. A cura di William W. Cohen e Andrew Moore. ACM, pp. 729–736 (cit. a p. 9).

Rizzoli, Andrea, Andrea Castelletti, Andrea Cominola, Piero Fraternali, A. Diniz dos Santos, Bruno Storni, Ricardo Wissman-Alves, Marco Bertocchi, J. Novak e I. Micheel (2014). «The SmartH2O project and the role of social computing in promoting efficient residential water use: a first analysis». In: *Proceedings of the 7th International Congress on Environmental Modelling and Software.* (Cit. a p. 27).

Russell, Stuart (1998). «Learning agents for uncertain environments». In: *Proceedings of the Eleventh Annual Conference for Computational Learning Theory*. A cura di ACM Press (cit. alle pp. VII, 1, 6, 7).

Sutton, Richard S. e Andrew G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press (cit. alle pp. VII, 1, 3–5).

Sutton, Richard S., David McAllester, Satinder Singh e Yishay Mansour (2000). «Policy gradient methods for reinforcement learning with function approximation». In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, pp. 1057–1063 (cit. a p. 10).

Syed, Umar, Michael H. Bowling e Robert E. Schapire (2008). «Apprenticeship learning using linear programming». In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML) 2008, Helsinki, Finland, June 5-9, 2008*. A cura di William W. Cohen, Andrew McCallum e Sam T. Roweis. ACM, pp. 1032–1039 (cit. a p. 8).

Syed, Umar e Robert E. Schapire (2007). «A Game-Theoretic Approach to Apprenticeship Learning». In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. A cura di John C. Platt, Daphne Koller, Yoram Singer e Sam T. Roweis. Curran Associates, Inc. (cit. a p. 8).

Tossou, Aristide C. Y. e Christos Dimitrakakis (2013). «Probabilistic inverse reinforcement learning in unknown environments». In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, Bellevue, WA, USA, August 11-15, 2013*. AUAI Press, Corvallis, Oregon (cit. a p. 10).

Van der Maaten, Laurens J.P. e Geoffrey E. Hinton (2008). «Visualizing High-Dimensional Data Using t-SNE». In: *Journal of Machine Learning Research* 9, pp. 2579–2605 (cit. a p. 42).

Ziebart, Brian D., Andrew Maas, J. Andrew Bagnell e Anind K. Dey (2008). «Maximum Entropy Inverse Reinforcement Learning». In: *Proceedings of the Twenty-Third Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. A cura di Dieter Fox e Carla P. Gomes. AAAI Press, pp. 1433–1438 (cit. a p. 9).