Modelli generativi di modelli neurali artificiali robusti

Uno studio preliminare di fattibilità

Emanuele Ballarin[†]

Relatore: Prof. Luca Bortolussi[‡]

Co-Relatore: Prof. Fabio Benatti[†◇]

[†]Dipartimento di Fisica, Univ. di Trieste
[‡]Dipartimento di Matematica e Geoscienze, Univ. di Trieste
[◇]INFN, sezione di Trieste

Sessione di Laurea straordinaria – 24 aprile 2020

Voi siete qui

Deep Learning $\subseteq$ *Machine Learning*

Il *Machine Learning* è essenzialmente studio e implementazione di **algoritmi d'apprendimento**. In particolare: apprendimento *statistico*.

> *"Un programma informatico impara da un'esperienza E con riferimento ad un'attività T e una misura di prestazioni P se le sue prestazioni misurate da P nello svolgimento dell'attività T migliorano noto E."*
>
> *(Mitchell, 1998)*

Nota

Più che *un* algoritmo, il *Deep Learning* è un approccio paradigmatico e universale alla risoluzione di problemi d'*apprendimento automatico*.

I problemi di *classificazione* automatica (*supervisionata*)

Spazio degli *input*: $\mathbb{I} \sim \mathbb{R}^k$ – Ciò che si vuole classificare
Spazio degli *output* (o *delle classi*): $\mathbb{O}$ – Classi in cui **partizionare** $\mathbb{I}$

Training set: $\{\mathbb{I} \times \mathbb{O}\} \supseteq \mathcal{T} = \{(\boldsymbol{x}_1, \xi_1), (\boldsymbol{x}_2, \xi_2), \ \ldots, \ (\boldsymbol{x}_n, \xi_n)\}$ – Esempi

Un classificatore $\mathcal{C}_{\boldsymbol{w},\boldsymbol{h}} : \ \mathbb{I} \to \mathbb{O}$ dipende da **pesi** ($\boldsymbol{w}$) e **iperparametri** ($\boldsymbol{h}$).
Loss function: $\mathcal{L} = \mathcal{L}(\mathcal{C}_{\boldsymbol{w},\boldsymbol{h}}, \mathcal{T})$

Scopo del *training*: trovare $\boldsymbol{w}$ ottimale per $\mathcal{C}$, noti $\mathcal{T}, \boldsymbol{h}, \mathcal{L}$

Questo è equivalente al problema di ottimizzazione:

$$\text{Find } \tilde{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\text{argmin}} \left(\mathcal{L} \left(\mathcal{C}_{\boldsymbol{w},\boldsymbol{h}}, \mathcal{T} \right) \right)$$

Il *Perceptron (Rosenblatt, 1958)*

Il *Perceptron* è un modello che opera sugli *input* con una **trasformazione affine** seguita da una *non-linearità* (A)

$$y = A\left(\boldsymbol{x} \cdot \boldsymbol{w} + b\right) = A\left(\sum_{j=1}^{k} x_j w_j + b\right)$$

e dotato di un semplice *training step* iterativo:

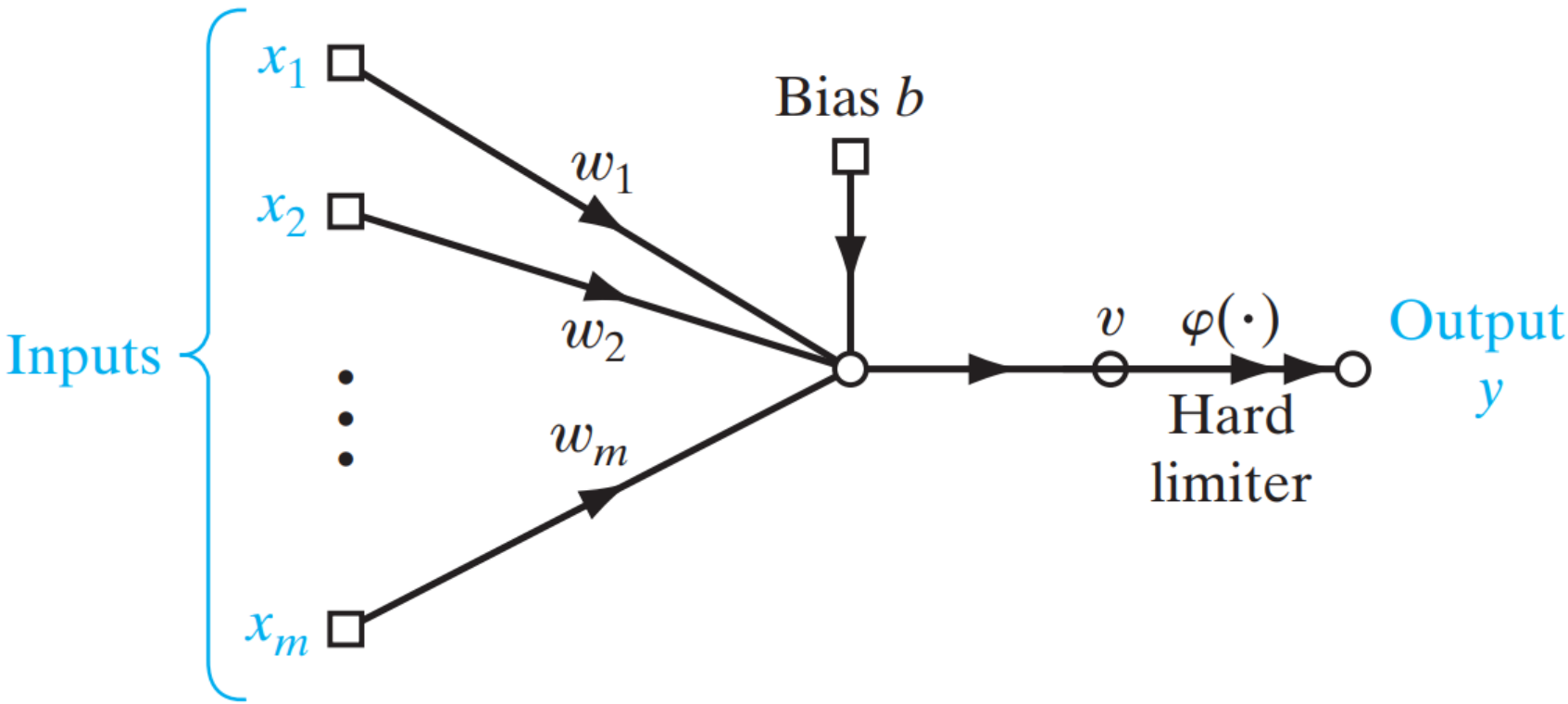
$$\boldsymbol{w'} \leftarrow \left(\boldsymbol{w} + \epsilon\left(\xi_i - y_i\right)\boldsymbol{x}_i\right)$$

$$b' \leftarrow \left(b + \epsilon\left(\xi_i - y_i\right)\right)$$

Nota

Il *Perceptron* non è solo un *classificatore*. È però in grado di risolvere solo una ristretta classe di problemi. *(Minsky & Papert, 1696)*

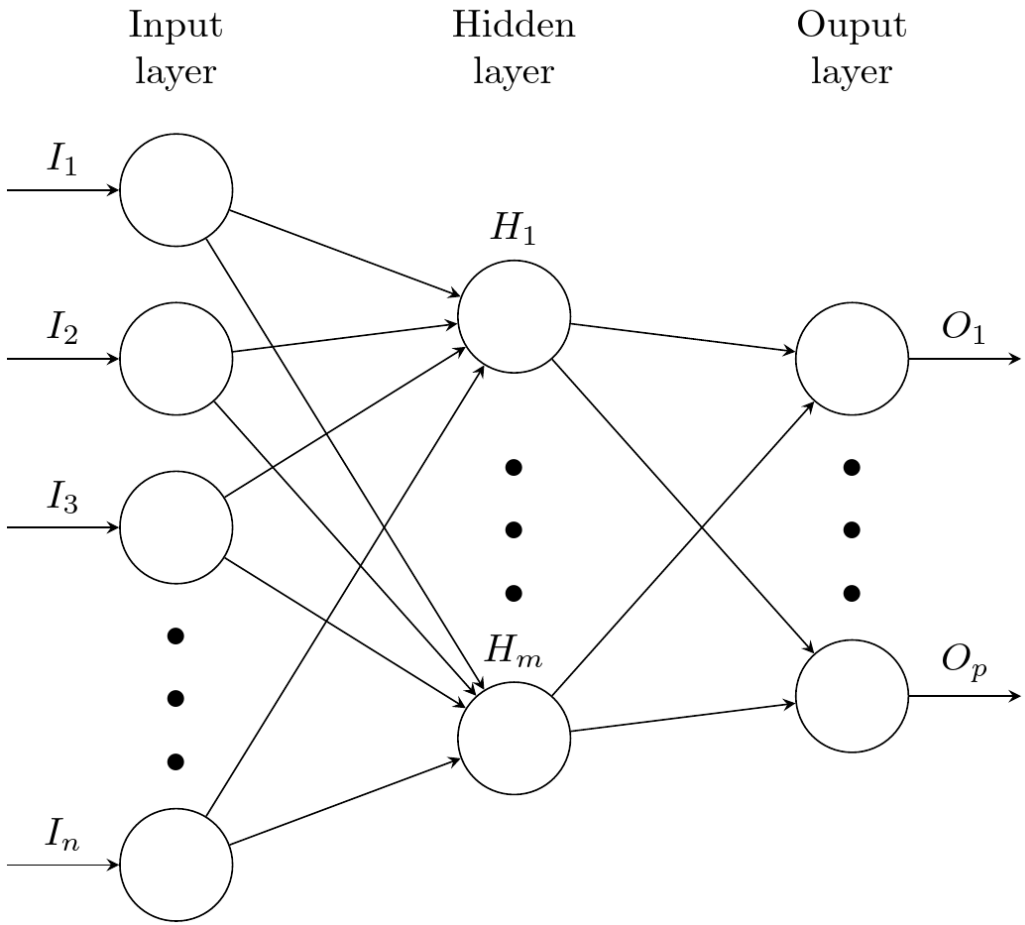
Il *Perceptron* (Rosenblatt, 1958) [cont.]



Going deeper

What if. . . ?

E se un problema d'apprendimento (arbitrario) fosse sempre **scomponibile** in sottoproblemi *perceptron-separabili*?



Going deeper [cont.]

Questo introduce una classe di modelli assai più *espressivi* e *capaci*: i *fully-connected multilayer perceptron models*.

E il concetto di *layer*

$$L_r(\boldsymbol{x}_r) = A_r\left(\boldsymbol{W}_r \boldsymbol{x}_r + \boldsymbol{b}_r\right)$$

.

Un'intera rete neurale può essere descritta come:

$$\boldsymbol{y} = \mathrm{NNet}(\boldsymbol{x}) = L_N(L_{N-1}(\ldots(L_1(\boldsymbol{x}))))$$

.

Going deeper [cont.]

Il problema d'ottimizzazione diventa però intrattabile con tecniche tradizionali. Si estende quindi l'approccio iterativo: *gradient descent*.

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

(con $\boldsymbol{\theta}$ vettore dei pesi del modello, $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathcal{L}$, $\epsilon \ll 1$)

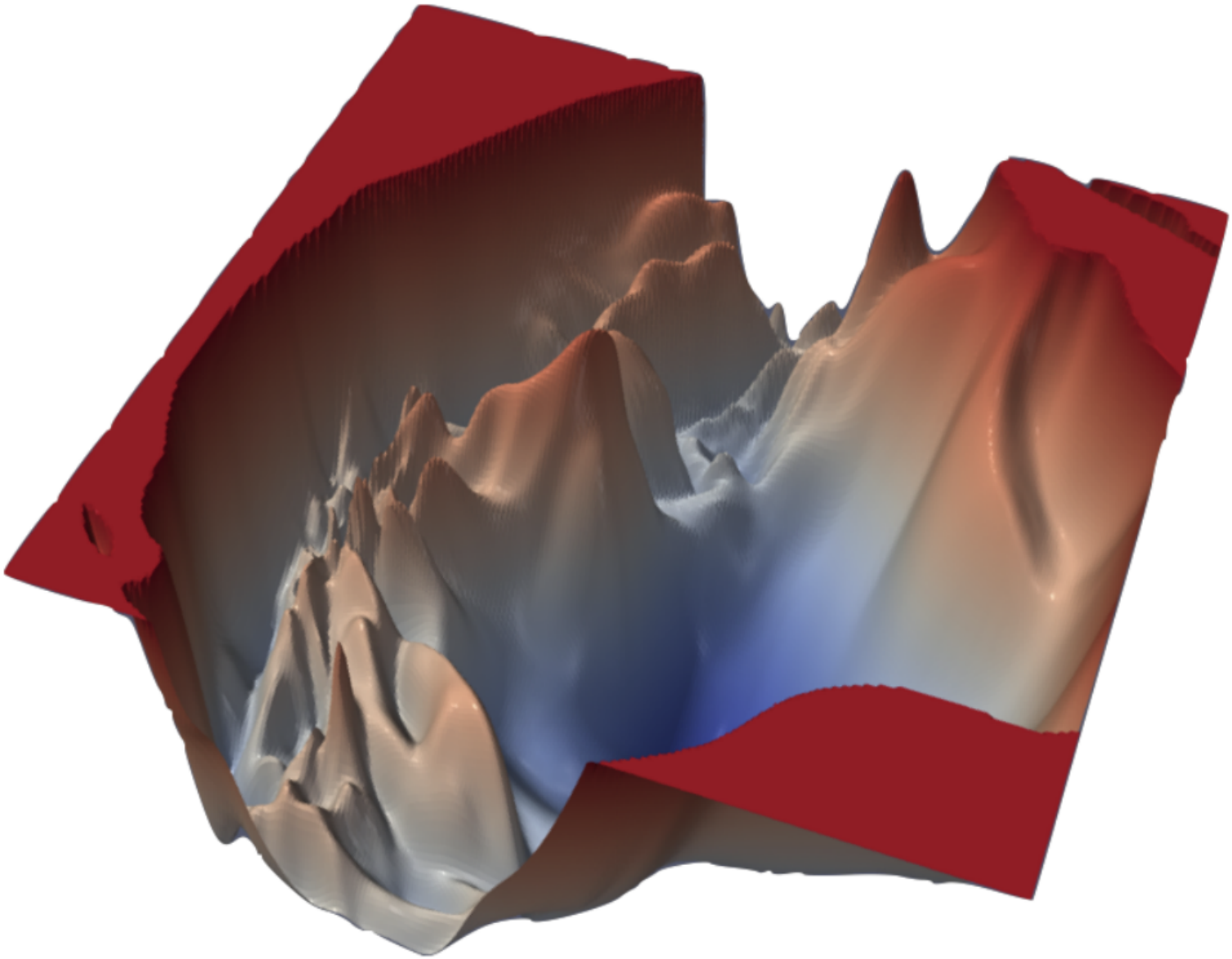
O sue varianti più efficaci (*g.d. with* 1^{st} *momentum correction*):

$$\boldsymbol{v}_t \leftarrow \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$$

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \epsilon \boldsymbol{g}_t + \mu \boldsymbol{v}_t$$

(con t l'indice di iterazione, $\mu \ll 1$)

Going deeper [cont.]



I risultati del *Deep Learning*

Il *Deep Learning* è un paradigma maturo, capace di risultati tutt'ora ineguagliati in problemi di *regressione predittiva*, *classificazione*, *generazione* di dati, *controllo*.

97.3% macaw

Tutto chiaro?

Forse no.

88.9% bookcase

(stesso classificatore; immagine quasi identica)

Voi siete qui

Perché studiare la *robustezza*?

Il fenomeno appena mostrato (*Perdikaris, 2018*) è un esempio di *assenza di robustezza* nel classificatore utilizzato.

Lo studio di questi fenomeni si presta ad analizzare:

- La validità del classificatore in scenari di **difficile prevedibilità**, rari, inusuali;
- La resistenza del classificatore a manipolazioni **dolose** degli *input*;

E di adottare eventuali mitigazioni.

Ma cosa provoca questo tipo di comportamenti?

La *Manifold Hypothesis (Dube, 2018)*

Un'ipotesi da tempo circolante all'interno della comunità dei ricercatori, ma solo recentemente associata ai fenomeni di *robustezza*.

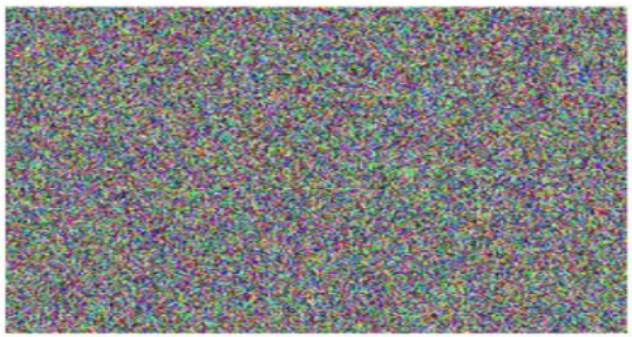
- Un classificatore (anche non *neurale*) accetta **qualsiasi** *input* compatibile con la codifica scelta.
- Il contenuto del *training set* descrive una varietà **basso**-*dimensionale* immersa in un *input space* **alto**-*dimensionale*.
  Ciò resta vero anche per le *decision boundaries* apprese.
- Lo scopo del *training* è quello di apprendere le **intersezioni** tra *data manifold* e *decision boundaries*;

Adversarial attacks

Un *adversarial attack* è una procedura (o il suo risultato) atta a provocare un comportamento non previsto o non voluto in un classificatore – cioè a produrre una *misclassification*.

Due approcci:

- **Perturbazioni** da *input* d'interesse (esempio precedente);
- *Input* privi di significato noto (vedi sotto).



(*Taj Mahal*; $\sim 65\%$)

Misure di *robustezza perturbativa*

Il *caso perturbativo* è sicuramente quello più simile a *input* imprevisti raccolti in uno scenario realistico o a causa di manomissioni.

ϵ-*robustezza* – di $\mathcal{C}$, in $\boldsymbol{x} \in \mathbb{I}$, rispetto a $||\cdot||$

$$\forall \boldsymbol{p} \text{ t.c. } ||\boldsymbol{p}|| < \epsilon,\ \mathcal{C}(\boldsymbol{x}) = \mathcal{C}(\boldsymbol{x} + \boldsymbol{p})$$

Empirical Global Robustness* – di $\mathcal{C}$, dati $\tilde{\mathcal{T}} \subseteq \mathcal{T}$ e un *attacco

$$\text{EGR}(\tilde{\mathcal{T}}) = \left(1 - \frac{\#(\text{attacks leading to misclassification})}{|\tilde{\mathcal{T}}|}\right)$$

Esempio di attacco: *PGD-*$||\cdot||_\infty$

È un attacco *white-box*: richiede **completa** conoscenza del modello $\mathcal{C}$ e della sua *loss* $\mathcal{L}$.

Dato $\tilde{\boldsymbol{x}} \in \mathbb{I}$ di corretta classificazione e fissato ϵ, **iterativamente**:

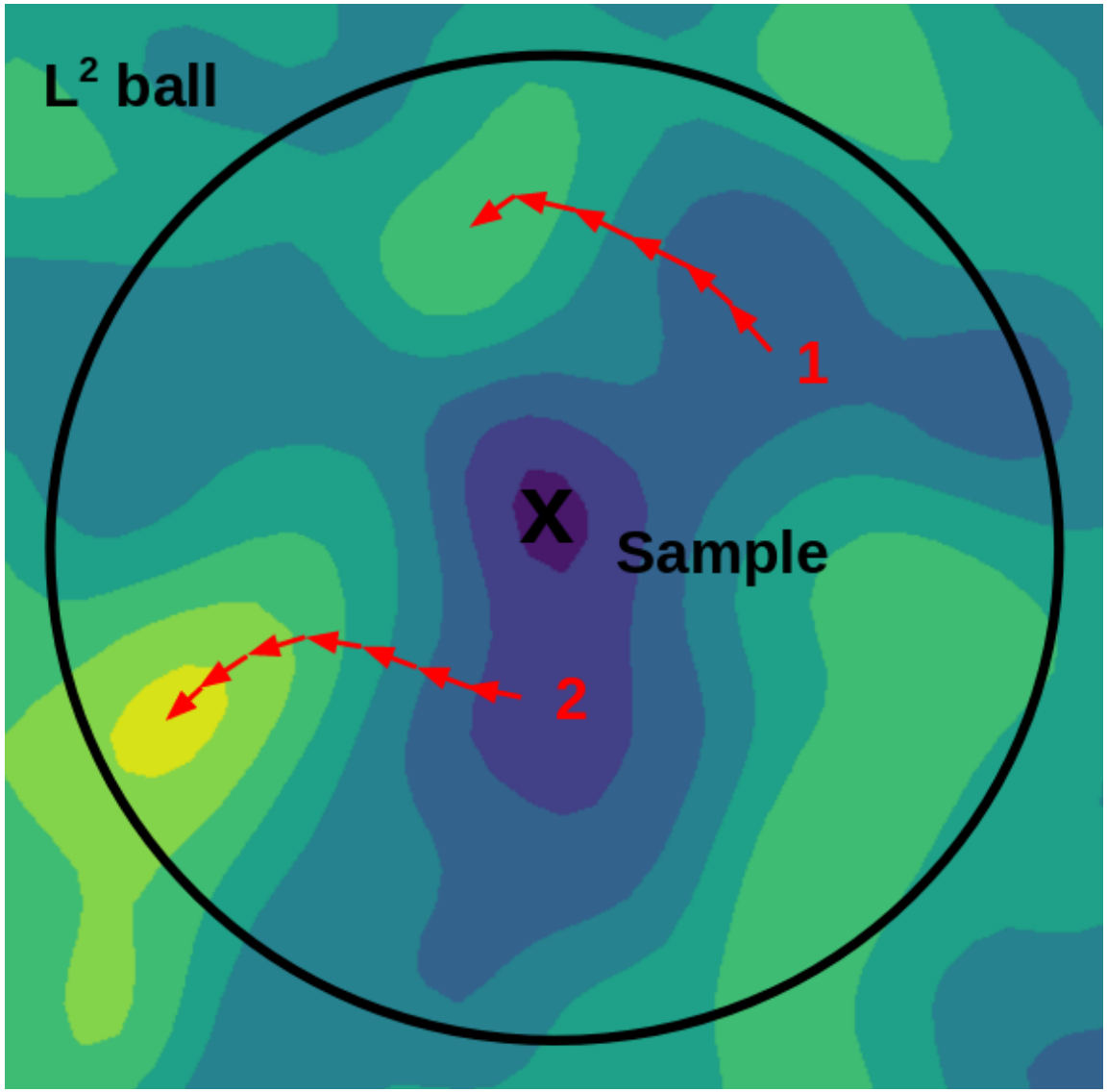
1. Ci si pone in $\tilde{\boldsymbol{x}}$ o in qualunque altro punto della palla chiusa $\mathcal{B}_\epsilon(\tilde{\boldsymbol{x}})$;
2. Si effettua un *update step*: $\boldsymbol{x'} \leftarrow \boldsymbol{x} + \eta \nabla_{\boldsymbol{x}} \mathcal{L}\left(\mathcal{C}_{\boldsymbol{w},\boldsymbol{h}}(\boldsymbol{x})\right)$; $\eta \ll 1$;
3. Qualora $\boldsymbol{x'} \notin \mathcal{B}_\epsilon(\tilde{\boldsymbol{x}})$, lo si proietta sulla sfera $\mathcal{S}_\epsilon(\tilde{\boldsymbol{x}})$.

È equivalente al problema di ottimizzazione:

$$\text{Find } \boldsymbol{x}' = \underset{\boldsymbol{x} \in \mathcal{B}_\epsilon(\tilde{\boldsymbol{x}})}{\text{argmin}} \left(-\mathcal{L}\left(\mathcal{C}_{\boldsymbol{w},\boldsymbol{h}}(\boldsymbol{x})\right)\right)$$

Le metriche scelte devono essere quelle indotte da $||\cdot||_\infty$.

Esempio di attacco: *PGD*-$||\cdot||_\infty$ [esempio nel caso $||\cdot||_2$]



Difese e *adversarial training*

In generale sono stati proposti numerosi *attacchi* per sistemi di *Deep Learning*. Così come per gli *attacchi* sono state proposte altrettante *difese*. Si cerca (e spesso si ottiene) la **trasferibilità**.

Un principio molto popolare è quello dell'*adversarial training*: proseguire l'allenamento su un *training set* contenente i risultati di *attacchi* generati allo scopo.

Nota

In generale, tuttavia, non sembra mai essere messo in discussione il meccanismo con cui i *pesi* debbano essere generati: tramite *gradient descent* sul modello stesso.

Voi siete qui

Modelli generativi dei pesi

"Nessuna metodologia di difesa contro adversarial examples è tuttavia completamente soddisfacente. Questo rimane un campo di ricerca aperto e in rapida evoluzione."

(Kurakin, Goodfellow & Bengio, 2018)

L'idea

Apprendere un modello (accompagnato da un'opportuno *protocollo di training*) con lo scopo di generare *pesi* di architetture neurali di forma prestabilita.

Generatore $\mathcal{G}$ e *value network* $\mathcal{V}$

Un riassunto della dinamica di *training*:

1. **Campionamento**: $\boldsymbol{s} \sim \text{Dist}^{z}$ nota;
2. Generazione dei pesi: $\boldsymbol{\theta} = \mathcal{G}(\boldsymbol{s})$ e *weight-loading*;
3. Computo di *accuratezza* $\mathfrak{A}$ e *robustezza* $\mathfrak{R}$ su $\tilde{\mathcal{T}} \subseteq \mathcal{T}$;
4. **Stima** di *accuratezza* e *robustezza* tramite $(\hat{\mathfrak{A}}, \hat{\mathfrak{R}}) = \mathcal{V}(\boldsymbol{\theta})$;
5. *Training step* per $\mathcal{V}$: $\mathcal{L}_{\mathcal{V}} =$ similarità tra $(\hat{\mathfrak{A}}, \hat{\mathfrak{R}})$ e $(\mathfrak{A}, \mathfrak{R})$
6. *Training step* $\mathcal{G}$: $\mathcal{L}(\boldsymbol{\theta}) = -\alpha\hat{\mathfrak{A}}(\boldsymbol{\theta}) - \beta\hat{\mathfrak{R}}(\boldsymbol{\theta})$ t.c. $\alpha + \beta = 1$.

Where's the catch?

In fase d'ideazione e sviluppo sono stati incontrati alcuni ostacoli che hanno richiesto soluzioni specifiche:

- Non-differenziabilità del *weight-loading* $\rightarrow$ Uso di un *value network* $\mathcal{V}$;
- Lentezza nella convergenza (a causa di $\mathcal{V}$ e non solo) $\rightarrow$ *pretraining*;

E alcune scelte sofferte ma necessarie:

- Volontà di preservare informazioni distribuzionali riguardo ai *pesi* $\rightarrow$ Impossibilità di usare *regolarizzazioni*;

Voi siete qui

Il dataset: *MNIST*

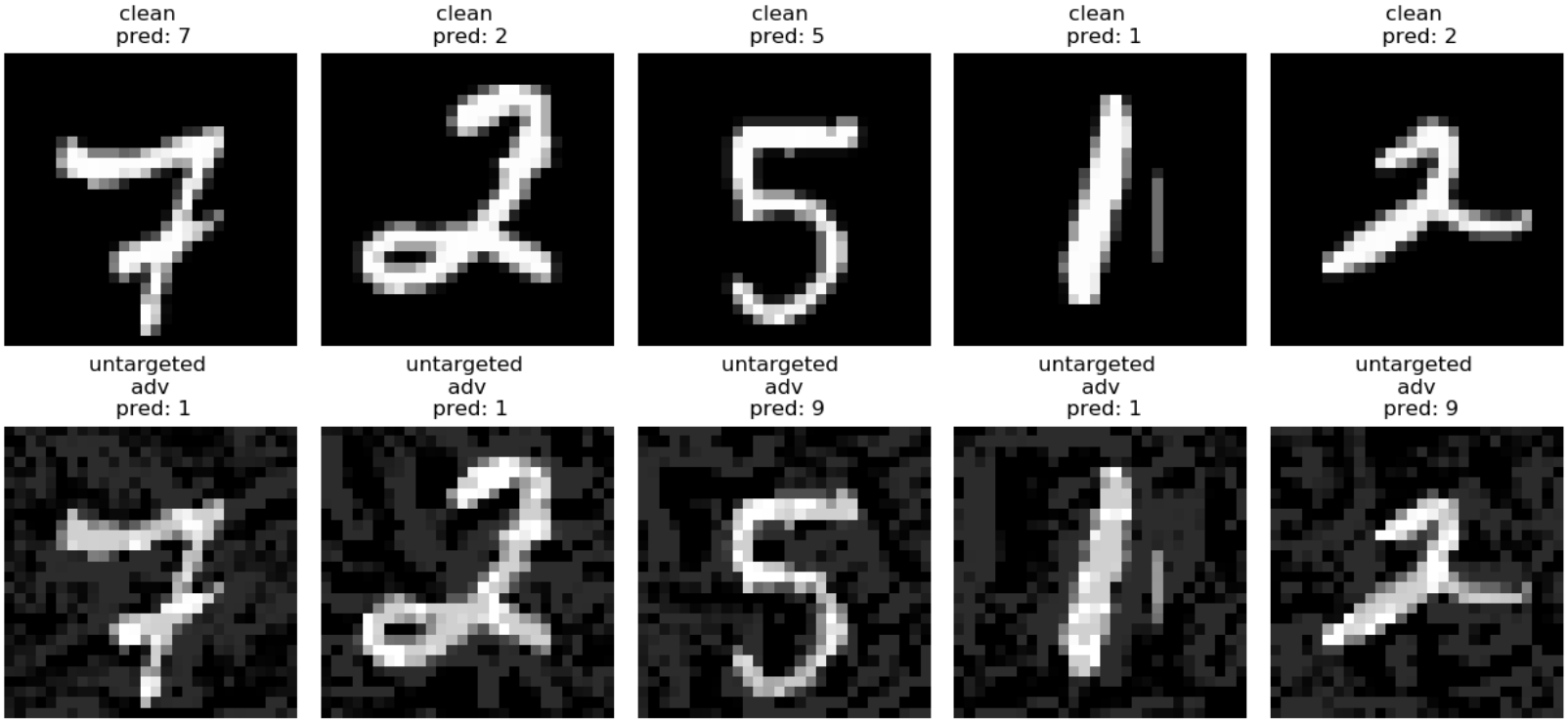
Raccolta di 10000 immagini in *bianco e nero*, quadrate, 28×28 pixel. Rappresentate come vettore binario dei 784 pixel.

Contiene raffigurazioni delle cifre arabe $0 - 9$ in diverse grafie manoscritte, con annessa classificazione in base alle intenzioni dello scrivente.

Tipico problema di classificazione: uno standard *de facto* per *toy-problems* in *Machine Learning*. Oggi di facile risoluzione quanto all'accuratezza del modello appreso.

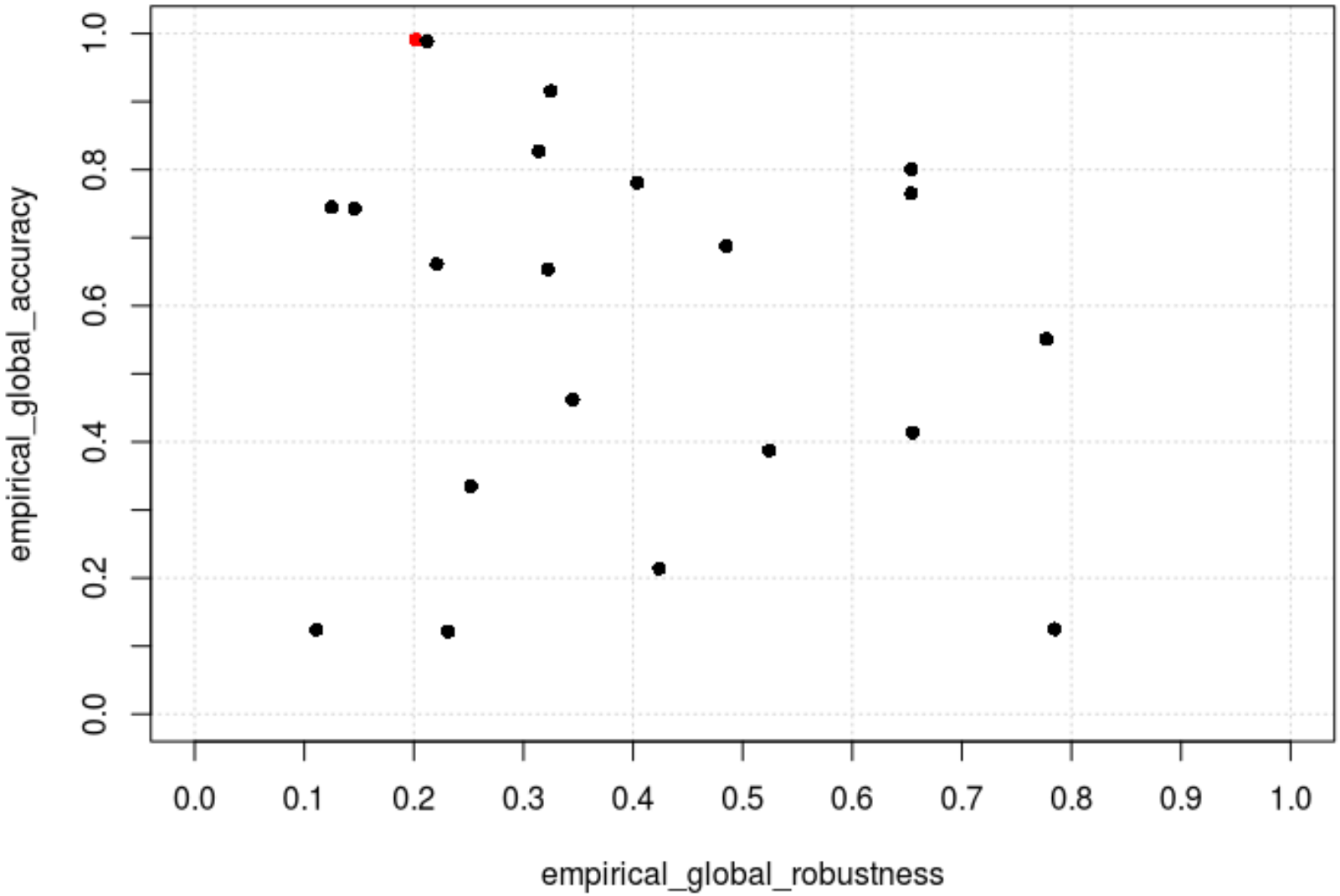
Classificatore (*LeNet-5*) e attacco (*PGD-*$||\cdot||_\infty$)

LeNet5 (LeCun, Bottou et al.) è un'architettura neurale *convoluzionale* e fully-connected pensata appositamente per la classificazione di cifre arabe manoscritte. Nel caso in esame è semplicemente stato ridotto per praticità il numero di *pesi*.



Risultati

Risultati interessanti, seppur fortemente preliminari.



Conclusioni

Alcuni dei risultati evidenziati si configurano come notevoli e interessanti.
Il percorso proposto è sicuramente meritevole di ulteriori approfondimenti.
L'ambito è ancora troppo poco esplorato!

Possibili sviluppi ulteriori:

- Determinazione delle condizioni rigorose di convergenza del modello;
- Studio dei campioni con migliore profilo *acc/rob*;
- Ottimizzazione multi-obiettivo nel *latent space*;
- Misure di *acc/rob* differenziabili e ablazione di $\mathcal{V}$;
- Robustezza *multi-attacco* e *multi-norma*;

- Robustezza *weight-agnostic*;
- Approccio *neuro-inspired/active learning* alla robustezza;

Ringraziamenti

Grazie per l'attenzione.