Input Similarity from the Neural Network Perspective

Guillaume Charpiat[1] Nicolas Girard[2] Loris Felardos[1] Yuliya Tarabalka[2,3]

[1] TAU team, INRIA Saclay, LRI, Université Paris-Sud. [2] Université Côte d'Azur, Inria. [3] LuxCarta Technology. [1,2,3] <firstname>.<lastname>@inria.fr

Introduction

- **Goal**: define and estimate the *similarity* of inputs, as perceived by the neural network
- **Motivation 1**: strong auto-denoising phenomenon in a multimodal image registration task (cf part IV)
  $\Longrightarrow$ accuracy far better than label noise!
  $\Longrightarrow$ analyze noise averaging effect over labels of *similar* examples
- **Motivation 2**: better understand neural network decisions
  $\Longrightarrow$ display examples considered as similar *by the network*

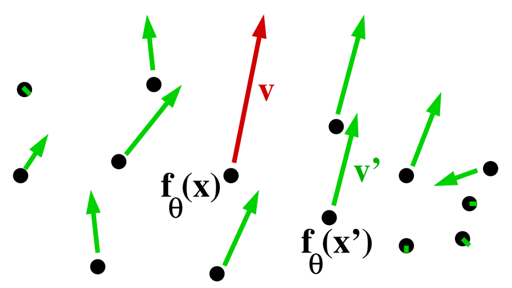
I - Building the similarity measure

Notations:

f_θ : trained neural network
θ : network parameters
$\mathbf{x}, \mathbf{x}'$: possible inputs

Similarity definition:

influence of $\mathbf{x}$ over $\mathbf{x}'$, quantified as:
how much an additional training step to push $f_\theta(\mathbf{x})$ in a certain direction would change the output for $\mathbf{x}'$ as well.



Output space

- If $\mathbf{x}$ and $\mathbf{x}'$ are very different (for the NN): changing $f_\theta(\mathbf{x})$ will barely affect $f_\theta(\mathbf{x}')$
- If $\mathbf{x}$ and $\mathbf{x}'$ are very similar: changing $f_\theta(\mathbf{x})$ will greatly affect $f_\theta(\mathbf{x}')$

- Changing $f_\theta(\mathbf{x})$ by a small quantity ε means updating θ by $\delta\theta = \varepsilon \dfrac{\nabla_\theta f_\theta(\mathbf{x})}{\|\nabla_\theta f_\theta(\mathbf{x})\|^2}$
- After update, new values for $\mathbf{x}$ and $\mathbf{x}'$:

$$f_{\theta+\delta\theta}(\mathbf{x}) = f_\theta(\mathbf{x}) + \nabla_\theta f_\theta(\mathbf{x}) \cdot \delta\theta + O(\|\delta\theta\|^2) = f_\theta(\mathbf{x}) + \varepsilon + O(\varepsilon^2)$$

$$f_{\theta+\delta\theta}(\mathbf{x}') = f_\theta(\mathbf{x}') + \nabla_\theta f_\theta(\mathbf{x}') \cdot \delta\theta + O(\|\delta\theta\|^2) = f_\theta(\mathbf{x}') + \varepsilon \underbrace{\frac{\nabla_\theta f_\theta(\mathbf{x}') \cdot \nabla_\theta f_\theta(\mathbf{x})}{\|\nabla_\theta f_\theta(\mathbf{x})\|^2}}_{\text{influence of } \mathbf{x} \text{ over } \mathbf{x}'} + O(\varepsilon^2)$$

- Symmetric kernel bounded in $[-1, 1]$: $k^C_\theta(\mathbf{x}, \mathbf{x}') = \dfrac{\nabla_\theta f_\theta(\mathbf{x})}{\|\nabla_\theta f_\theta(\mathbf{x})\|} \cdot \dfrac{\nabla_\theta f_\theta(\mathbf{x}')}{\|\nabla_\theta f_\theta(\mathbf{x}')\|}$

Properties for vanilla neural networks

Theorem. *For any real-valued network f_θ without parameter sharing, if $\nabla_\theta f_\theta(\mathbf{x}) = \nabla_\theta f_\theta(\mathbf{x}')$ for two inputs $\mathbf{x}, \mathbf{x}'$, then all useful activities computed when processing $\mathbf{x}$ are equal to the ones obtained when processing $\mathbf{x}'$.*

Rewriting: $k_\theta(\mathbf{x}, \mathbf{x}') = \displaystyle\sum_{\text{activities } i} \lambda_i(\mathbf{x}, \mathbf{x}')\, a_i(\mathbf{x})\, a_i(\mathbf{x}')$ where $\lambda_i(\mathbf{x}, \mathbf{x}') = \displaystyle\sum_{\text{neuron } j \text{ using } a_i} \frac{df_\theta(\mathbf{x})}{db_j} \frac{df_\theta(\mathbf{x}')}{db_j}$

$\Longrightarrow$ data-dependent importance weights, vs. $\lambda_i(\mathbf{x}, \mathbf{x}') = \lambda_{\text{layer}(i)}$ for the *perceptual loss*

General case (parameter-sharing networks)

properties above do not hold anymore, as invariances (s.t. as translation) are incorporated in the similarity

Higher output dimension

- $f_\theta(\mathbf{x}) = \big(f^i_\theta(\mathbf{x})\big)_{i \in [1,d]} \in \mathbb{R}^d$ with $d > 1$
- $K_\theta(\mathbf{x}', \mathbf{x})$ the $d \times d$ kernel matrix defined by $K^{ij}_\theta(\mathbf{x}', \mathbf{x}) = \nabla_\theta f^i_\theta(\mathbf{x}') \cdot \nabla_\theta f^j_\theta(\mathbf{x})$
- Unitless symmetrized normalized kernel: $K^C_\theta(\mathbf{x}, \mathbf{x}') = K_\theta(\mathbf{x}, \mathbf{x})^{-1/2}\, K_\theta(\mathbf{x}, \mathbf{x}')\, K_\theta(\mathbf{x}', \mathbf{x}')^{-1/2}$
- Similarity in a single value: $k^C_\theta(\mathbf{x}, \mathbf{x}') = \frac{1}{d} \operatorname{Tr} K^C_\theta(\mathbf{x}, \mathbf{x}')$.
- Other metrics possible in output space
  e.g. for rotation invariance: $k^{C,\text{rot}}_\theta(\mathbf{x}, \mathbf{x}') = \dfrac{1}{2} \sqrt{\left\|K^C_\theta(\mathbf{x}, \mathbf{x}')\right\|^2_F + 2 \det K^C_\theta(\mathbf{x}, \mathbf{x}')}$

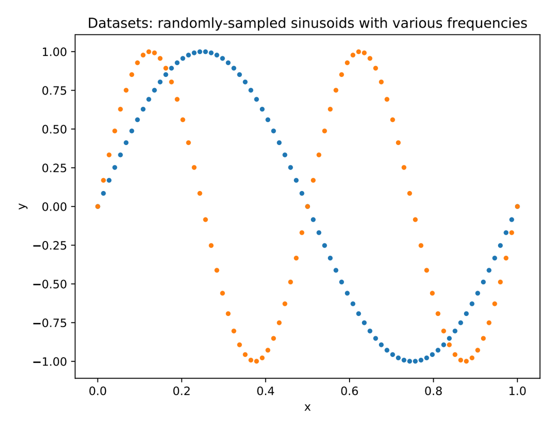
II - Estimating density

How many samples $\mathbf{x}'$ are similar to $\mathbf{x}$ according to the network? Many ways to count!

hard-thresholding with threshold $\tau \in [0, 1]$	soft estimate **Fast!** $O(NP)$	less-soft positive-only estimate $(\alpha > 0)$
$\sum_{\mathbf{x}'} \mathbb{1}_{k^C_\theta(\mathbf{x},\mathbf{x}') \geqslant \tau}$	$\sum_{\mathbf{x}'} k^C_\theta(\mathbf{x}, \mathbf{x}') = \frac{\nabla_\theta f_\theta(\mathbf{x})}{\|\nabla_\theta f_\theta(\mathbf{x})\|} \cdot \frac{\overline{\nabla_\theta f_\theta}}{\|\overline{\nabla_\theta f_\theta}\|}$	$\sum_{\mathbf{x}'} \mathbb{1}_{k^C_\theta(\mathbf{x},\mathbf{x}')>0}\, k^C_\theta(\mathbf{x}, \mathbf{x}')^\alpha$

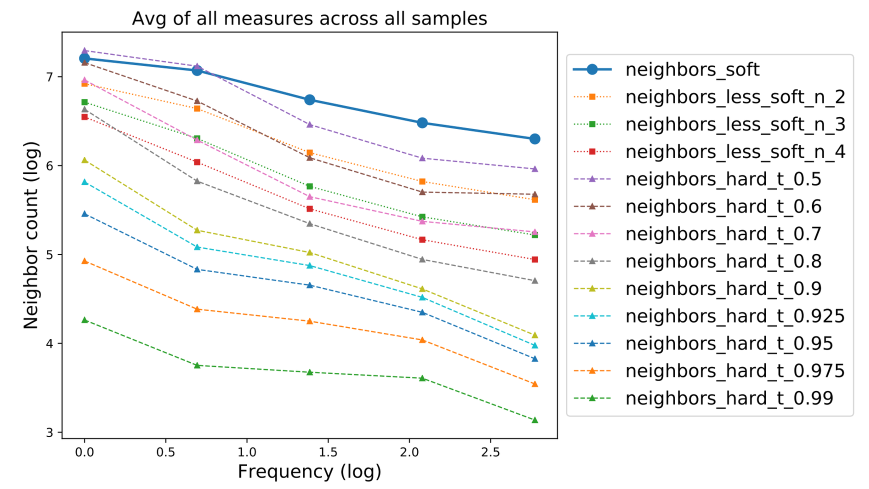
Toy dataset:

- Task f : learn to reproduce $\sin(2\pi f x)$, from $N = 2048$ random samples
- increase frequency $f \Longrightarrow$ fewer neighbors

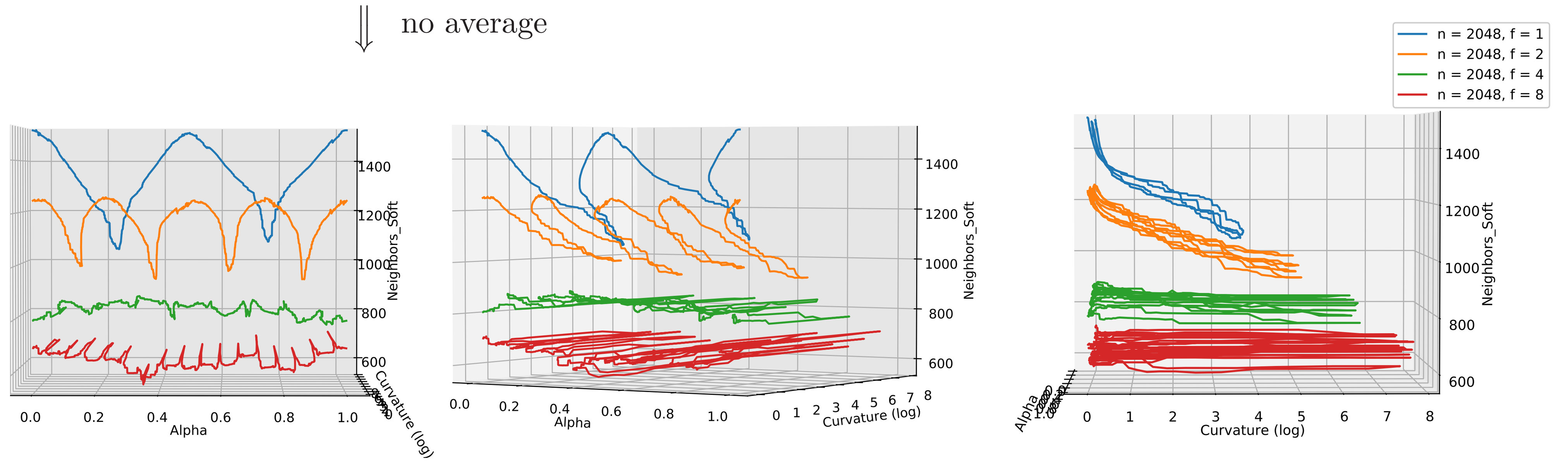


average per task

$\Longrightarrow$



$\Downarrow$ no average



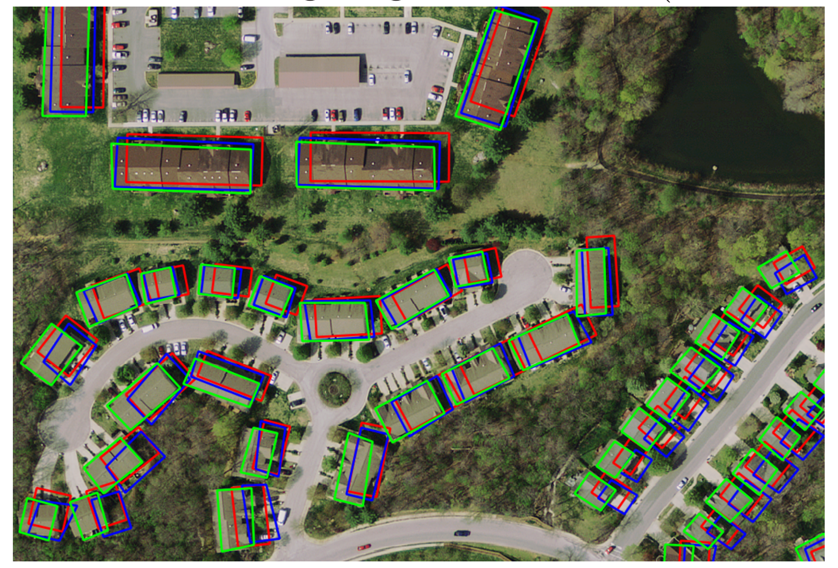
Soft estimate of the number of neighbors as a function of location and curvature

III - Futher possible uses of similarity

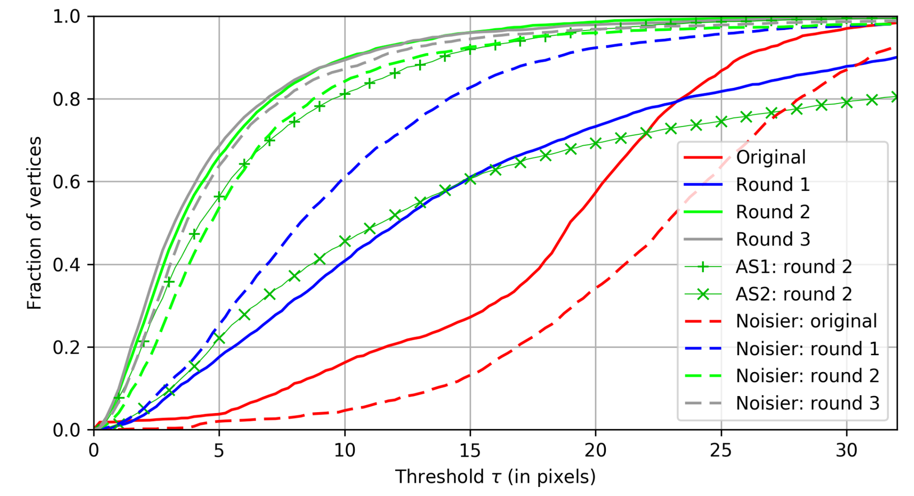
- similarity computation $\Longrightarrow$ density estimator $\Longrightarrow$ **active learning**
  $\Longrightarrow$ + local variance (of predictions / of target labels): **overfit/underfit detection**
- **uncertainty** estimation: how much should the training set be changed to change the prediction $f_\theta(\mathbf{x})$ at point $\mathbf{x}$?
  $\Longrightarrow$ sum influence factors from all training points $\mathbf{x}_i$ $\Longrightarrow$ **reliability** $\propto \sum_i k_\theta(\mathbf{x}, \mathbf{x}_i) \,/\, \|\nabla_\theta f_\theta(\mathbf{x})\|^2$
- all quantities are **differentiable** $\Longrightarrow$ enforce similarity properties during training!
  - incite the network to consider given $\mathbf{x}$ and $\mathbf{x}'$ as (dis)similar : add $(\pm)\, k_\theta(\mathbf{x}, \mathbf{x}')$ to the loss
  - asking a subset $\mathcal{S}$ of samples to be treated as similar :
    $\sum_{i,j \in \mathcal{S}, i \neq j} k^C_\theta(\mathbf{x}_i, \mathbf{x}_j)$ rewrites as a function of $\mu = \frac{1}{n} \sum_{i \in \mathcal{S}} \frac{\nabla_\theta f_\theta(\mathbf{x}_i)}{\|\nabla_\theta f_\theta(\mathbf{x}_i)\|}$
  - **dynamics of learning :** speed-up effect noticed when training on MNIST with class-similarity enforcement

IV - Analysis of self-denoising phenomena

Multimodal image registration task (RGB image to cadaster map) from noisy labels [1]



Red: initial dataset annotations (noisy)
Blue: prediction after learning from Red
Green: prediction after learning from Blue



Accuracy cumulative distributions (ground truth from [2]), i.e. fraction of pixels whose registration error $\leqslant$ abscissa.

$\Longrightarrow$ impressive self-denoising effect. Explanation from Noise2Noise [3]:
same example x showed N times with noisy labels $\sim \mathcal{N}(y, \sigma_\varepsilon)$ $\Longrightarrow$ best prediction $\widehat{y} = y \pm \frac{1}{\sqrt{N}} \sigma_\varepsilon$
$\Longrightarrow$ noise averaging effect over labels of *similar* examples?

Formalism:

input : $\mathbf{x}_i$
true (unknown) label : y_i
(unknown) noise : ε_i (iid, centered)
noisy (available) label : $\widetilde{y}_i = y_i + \varepsilon_i$
predicted label : $\widehat{y}_i = f_\theta(\mathbf{x}_i)$
training loss : $L(\theta) = \sum_j ||\widehat{y}_j - \widetilde{y}_j||^2$

$\Longrightarrow$

- at convergence $\nabla_\theta E = 0 \Longrightarrow \mathbb{E}_k[\widehat{y}] = \mathbb{E}_k[\widetilde{y}]$
- $\mathbb{E}_k[a] := \sum_j a_j\, k_\theta(\mathbf{x}_i, \mathbf{x}_j)$: mean value around $\mathbf{x}_i$

$$\widehat{y}_i - \mathbb{E}_k[y] = \mathbb{E}_k[\varepsilon] + (\,\widehat{y}_i - \mathbb{E}_k[\widehat{y}]\,)$$

- $\widehat{y}_i - \mathbb{E}_k[y]$: prediction error to smoothed true labels
- $\mathbb{E}_k[\varepsilon] \propto \sigma_\varepsilon \, \|k_\theta(\mathbf{x}_i, \cdot)\|_{L2} \Longrightarrow$ denoising factor: 0.02
- Shift: $(\,\widehat{y}_i - \mathbb{E}_k[\widehat{y}]\,)$: 4.4 px

Similar patch retrieval, and comparison with *perceptual loss*:

Source | Closest neighbor patches

Discussion

- Fast similarity / density estimation opens the door to underfit/overfit/uncertainty analyses and control
- Extended Noise2Noise [3] to non-identical inputs: self-denoising effect as a function of inputs similarities
- Links with *Neural Tangent Kernel* [4]: same concept! used differently
- Future work: analyze and improve robustness to adversarial attacks
- Code available on GitHub: `http://github.com/Lydorn/netsimilarity` or scan the QR code:

References

[1] N. Girard et al. Noisy Supervision for Correcting Misaligned Cadaster Maps Without Perfect Ground Truth Data. In *IGARSS* 2019.
[2] E. Maggiori et al. Can semantic labeling methods generalize to any city? the Inria aerial image labeling benchmark. In *IGARSS* 2017.
[3] J. Lehtinen et al. Noise2noise: Learning image restoration without clean data. In *ICML* 2018.
[4] A. Jacot et al. Neural tangent kernel: Convergence and generalization in neural networks. In *NIPS* 2018.