Guarantees : generalization / robustness

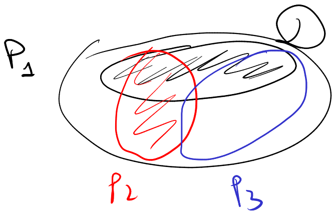
I Generalization

I-A) Ensemble methods

- agglomerate different predictors → different techniques : → SVM, → random forests, → deep learning
  - → same technique, different parameters → different kernels; → deep : different architectures / hyper-parameters
  - → same technique, same parameters → different parts of the dataset

↓ average all predictors or more advanced ways (use the validation set)

↳ simple case : $x \mapsto \underset{k\ \text{(predictors)}}{\sum} f_k(x) / \#k$



[Distilling the knowledge in a Neural Network] "Teacher-student" / "Distillation"

- hard task
- "small" network : training → poor performances

several " s : → " "

$\underbrace{\text{set of predictors}}$ ensemble method (average) → good performance

issue : many networks
↳ lot of space
↳ " " computational resources

train a "small" network g to imitate the ensemble method

task : $g \simeq \frac{1}{4} \sum_k f_k$

student →

← teacher

↳ this training works → good performance

Other ensemble techniques :

- boosting : combining weak classifiers into a strong one

wrongly classified samples (by f_t)

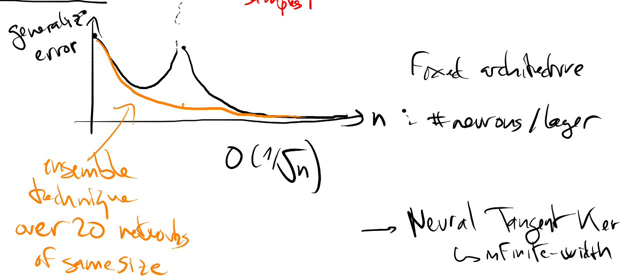
$f_t \longrightarrow f_{t+1}$: different weights to samples
- ↗ weight for wrongly-classified
↳ train a weak classifier with these weights
↳ agglomerate : $f_{t+1} = f_t + wc_t(x)$

I-B Generalization without regularization

↳ no L^2 penalty on weights
↳ no functional norm

→ loss : $\sum_{\text{samples } i} \|\hat{y}_i - y_i\|^2$ or cross-entropy

"double gradient descent"

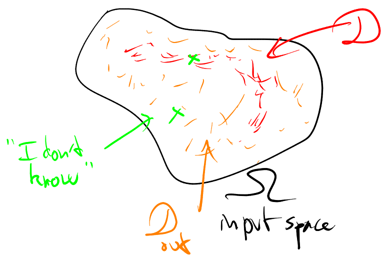


Fixed architecture

→ Neural Tangent Kernel
↳ infinite-width layers

I-C Generalize only when relevant

[Towards N.N. that provably know when they don't know] Matthias Hein's team



↳ use 2 distributions → $\mathcal{D}$: data "in" ex: CIFAR

→ $\mathcal{D}_{out}$: to model samples outside of $\mathcal{D}$ ex: ImageNet

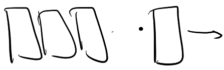
↳ kernels → "distance" to $\mathcal{D}_{in}$

→ " to $\mathcal{D}_{out}$

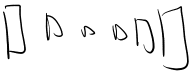
Training:
- samples in $\mathcal{D}_{in}$: loss (data loss) → $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ target (label)
- " in $\mathcal{D}_{out}$: loss: $\hat{y} \simeq \begin{pmatrix} 1/k \\ 1/k \\ 1/k \\ \vdots \end{pmatrix}$ uniform distribution over k classes

II Learning from noisy data

Denoising auto-encoder [ICML 2008]

- dealing with noisy data ≃ noise modeling
- originally: $x \to$  $\to \tilde{x}$ auto-encoder

usually: narrow middle layer to concentrate information
- set-up:

input: noisy version of x → x^{n} →  → x goal: reconstruct i.e. get rid of noise

↳ learn | more robust features
| to denoise

Classification with noisy labels [D.L. is robust to massive label noise; 2017]

- true distribution (x, y)
- given noisy " $(x, \tilde{y})$ → sometimes y → sometimes random labels

- if some samples are mislabeled (in the training set) → not much change of accuracy
- a significant part of - - - - - - - → ?

50%
99% → still possible to get reasonable results <u>provided</u> that data is available in large quantities

↳ what matters is the number of correct labels

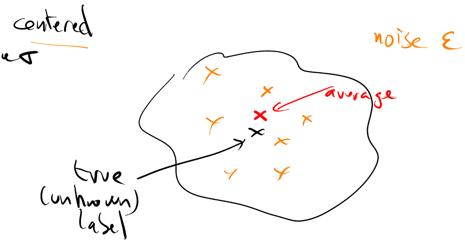
→ noise 50% → ×10 data

99% → ×100 data

Regression with noisy labels

- dataset $(x, y + \varepsilon)$

↳ noise iid centered variance σ
- simplistic case:
  - always the same input x



noisy target $\tilde{y}_i = y(x) + \varepsilon_i$

L^2 loss

$$\sum_i \| \underbrace{\hat{y}(x)}_{\text{always same prediction}} - \tilde{y}_i \|^2$$

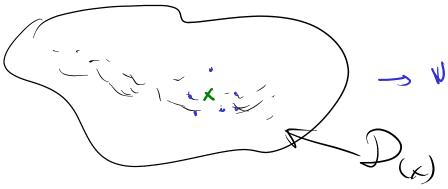
→ average of $\tilde{y}_i$

[Noise2Noise] denoising effect: $1/\sqrt{N}$ Factor $\Longleftarrow$ classical statistics

N samples $\Rightarrow |\mu - y| \sim \frac{1}{\sqrt{N}} \times \sigma$

$\uparrow \frac{1}{\#i} \sum_i \tilde{y}_i$ $\quad \uparrow$ true label

, real case : x varies



→ N.N. will perform an "average" over similar points

$$\simeq \frac{1}{\sqrt{\#\text{similar examples}}}$$

↳ define "similarity" according to the network: based on the ability of the N.N. to distinguish samples

↳ similarity between samples x and x' : $\frac{\nabla_\theta \hat{g}_\theta(x)}{\| \ " \ \|} \cdot \frac{\nabla_\theta \hat{y}_\theta(x')}{\| \ " \ \|} \in \mathbb{R}$ if $\hat{g} \in \mathbb{R}$

$\in [-1, 1]$

↳ how does label noise in training set affect prediction at a given test point?

↳ can compute a denoising factor

noise σ training → prediction noise : $\sigma /$ factor

↑ depends on the test sample

[Understanding black-box predictions via Influence functions, ICML 2017]

↳ based on classical statistics

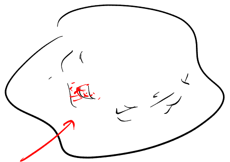
→ noise on x

→ noise on the sampling distribution : probability to pick x

↳ same kind of formulas

→ ∇ Loss instead of $\nabla \hat{g}$

→ inverse of Hessian



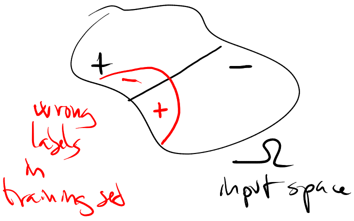
increase sampling rate in specific area

Classification :

[Learning with noisy labels]

↳ Hyp: know the amount of noise

eg: 20% of labels are inverted (binary classific°)



→ build a loss : unbiased

↳ on average, correct loss! as if with true labels

$\mathbb{E}$ samples

→ variance?

Rademacher bound !

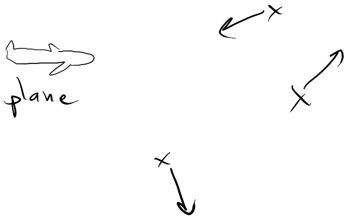
↳ notion of complexity

III Formal proofs

- to prove (formally) that the N.N. will <u>never</u> mistake
- express a property → check it

$\exists! \ \forall$ input samples $x \in \mathcal{X}$, $|\hat{y}(x) - y(x)| \leq \varepsilon$ completely specified

- ACAS-Xu :



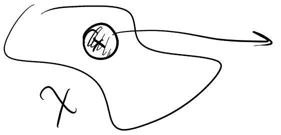
given: - locations
- speeds

(no perception)

prove: no collision

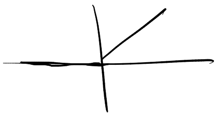
small network
3 inputs? → few outputs
1000 neurons

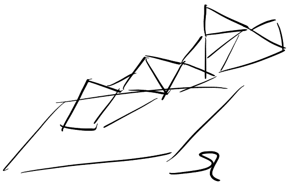
- local properties: locally robust to adversarial attacks



no adversarial ex in that ball

- key words: - abstraction
  - ReLuplex → N.N. with ReLU
    ↳ piecewise-affine f°





- naive application of traditional provers → complexity ~ $2^{\#\text{neurons}}$