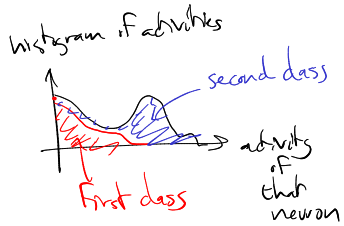
Chapter 2 : Interpretability

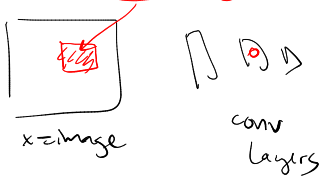
I Visualizat°/Analysis (of a trained network)

At the neuron level

- pick one neuron : activities on the training set → stats
  ex: classif° task

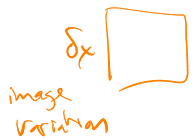


- what it sees → receptive field



- what does it react to?
  - → display input patterns that maximize the activity
    - ↳ from the training set
  - → compute the pattern that would " " " "
    - by gradient descent: $\frac{\partial x_t}{\partial t} = \eta \frac{\partial \text{activity}(x_t)}{\partial x}$ (↑ gradient)

↳ if you apply this to the full input, looking at output neurons:
e.g. classific° task,



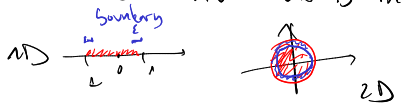
$\frac{\partial p(\text{class } c)}{\partial \text{image } x}$: sensitivity of the prediction (for class c) to the input x

$x' = x + \eta \frac{\partial p(\text{other class})}{\partial x}$ (η not much) ⇒ change completely the prediction

↳ adversarial examples [2014]
↳ adversarial attacks
↳ due to data dimension

high-dim Data space

- no dense sampling (requires $10^{\#\text{dim}}$ samples)
- all points are on the boundary
  ↳ measure concentration
  → look at uniform distrib° in the unit-ball



proportion of points close to the boundary → 1 (dim → ∞)

robustify

↳ train with $x \in$ training set ↦ ℓ
and $x + \delta x$ associated adversarial attack ↦ ℓ

≃ smooth function: $\frac{\partial f_{\ell,x}}{\partial x}(\delta x) = 0$

↳ $\left\| \frac{\partial f}{\partial x} \right\|^2$

adversarial attacks ⇄ robustifying techniques (loop)

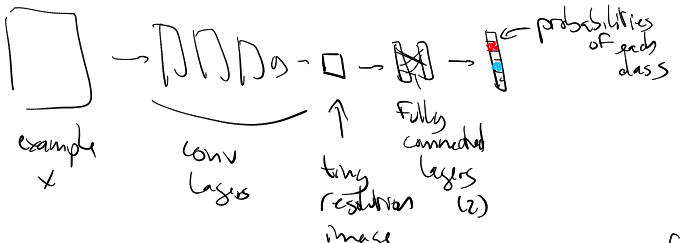
- does it have an impact? $\frac{\partial f}{\partial a}(x)$ ← activity of that neuron → find which neurons influence the output the most

⇒ reasoning at the neuron level: compare to human brain

The case of CNN

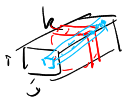
- which parts of the image are responsible for the decision?

grad-CAM: Class Activation Maps → classif° pb



- pick a class c

↳ activities of that layer: A^k_{ij} ← feature, ← pixel location



$\longmapsto p(c) = y_c$

- importance of feature k for class c:

$$\alpha^c_k = \frac{1}{\#\text{pixels}} \sum_{ij} \frac{\partial y_c}{\partial A^k_{ij}} \in \mathbb{R}$$

- importance of one pixel (i,j):

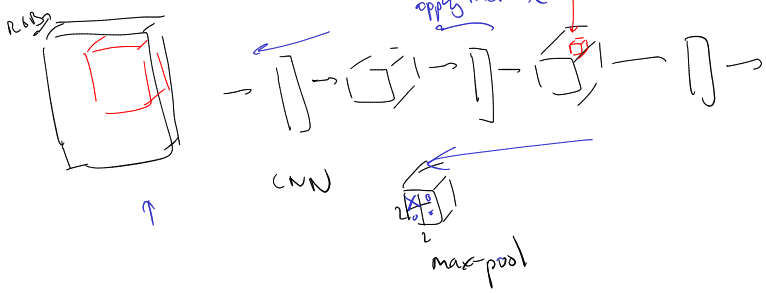
$$\sum_k \alpha^c_k A^k_{ij} \in \mathbb{R}$$

$$-\text{ReLU}\left(\sum_k \alpha^c_k A^k_{ij}\right)$$

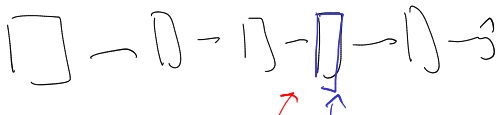
⇓ heat map for class c

on the tiny resolution → Scale it up to the original resolution → Blurry

- visualize the filters that are learned



At the layer level



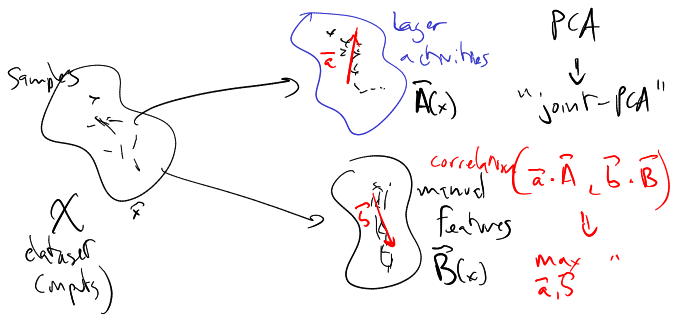
all activations Correlations? → CCA

Canonical Correspondence Analysis

manual list of physically meaningful features

$\phi_k(x)$

vector representing the input



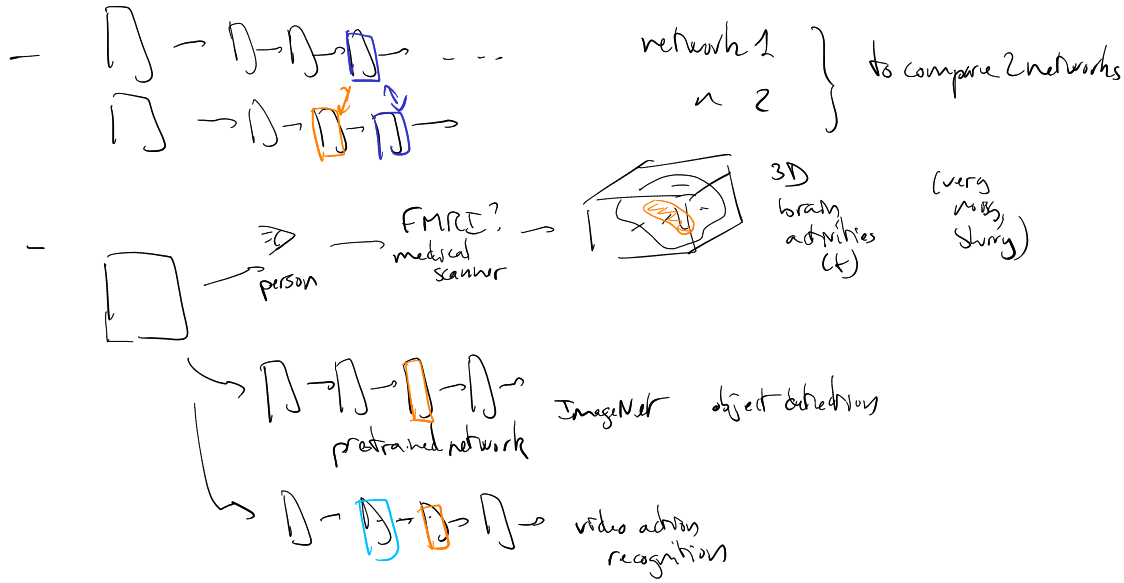
nonlinear → CKA

Linear tools

PCA

↓

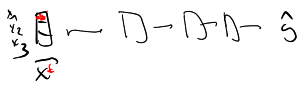
"joint-PCA"



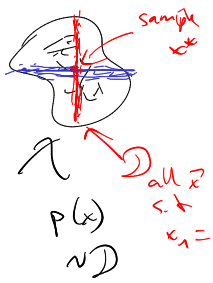
Analyzing inputs

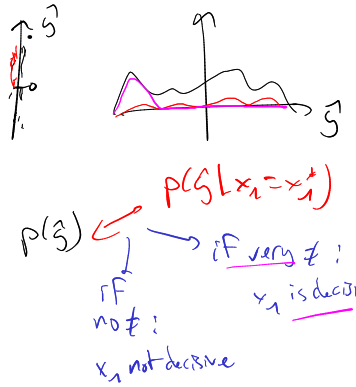
- which parts of the input are responsible for this decision (for that particular sample)
  - ↳ low-dim input:

Shapley values

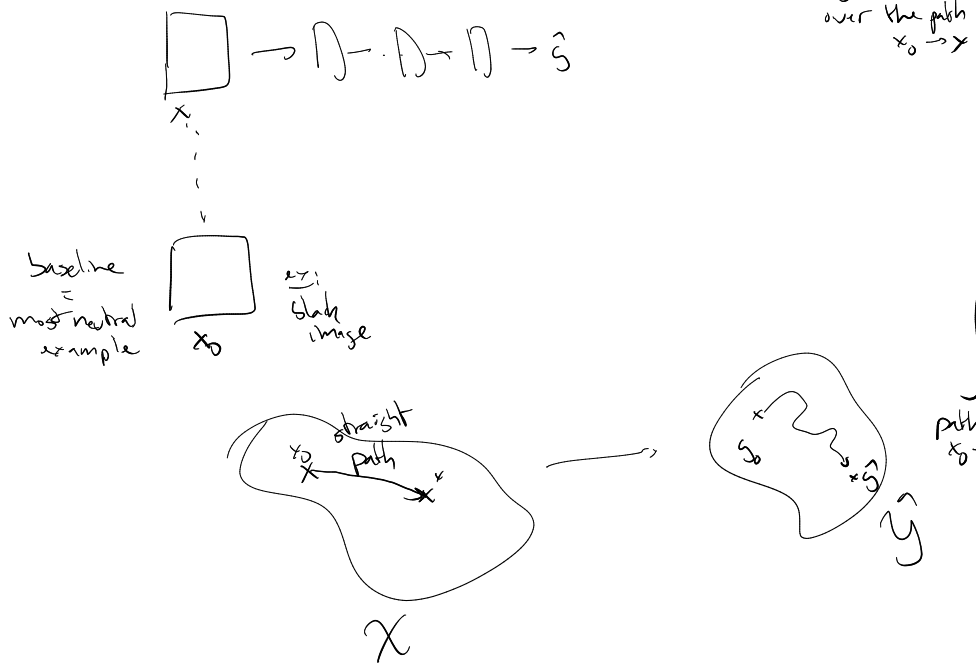


$p(\hat{S}, x)$

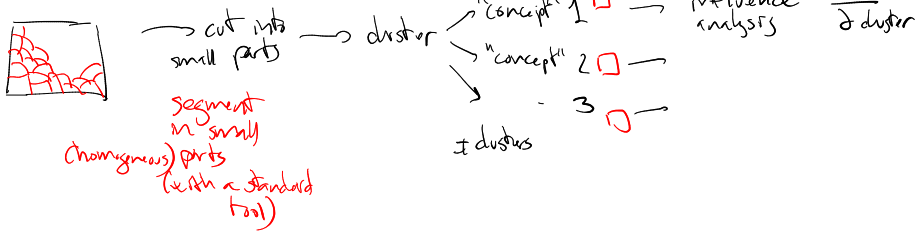




- path gradient: $\frac{\partial \hat{S}}{\partial x}$: issue → cf. adversial example
  - ↳ "looks weird" ⇒ regularize by integrating over the path $x_0 \to x$

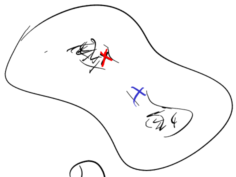


- which parts of the inputs / which patterns are statically meaningful?
  - CAV / ACE : automatic concept-based explanations display cluster center



- which input samples are similar, or influent?

- similarity:



$\mathcal{D} = \mathcal{X}$

which metric?

↳ some meaningful metric in $\mathcal{X}$ → hand-designed

contre-ex: L^2 metric between images

↳ according to the network

↳ similar outputs?

↳ log-proba vector (p(each class))

↳ similar activities

"perceptual loss"

$x \to \square \to \square \to \square \to \hat{y}$

$\vec{A}_l(\vec{x}) = $ [vector]

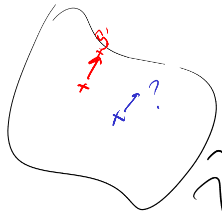
$\vec{A}_l(\vec{x'}) = $ [vector]

L^2 norm → metric

weight for each layer

$\sum_l \alpha_l \|\vec{A}_l(x) - \vec{A}_l(x')\|^2$

(preselected layers)



$\hat{\mathcal{Y}}$

VGG network

ResNet

- influence functions:

which samples in the training set are responsible for the decision taken for $\vec{x}$?

→ first idea: remove 1 sample $\vec{x'}$ (ablation study)

& retrain

& see if difference for $\vec{x}$

⇒ too computationally demanding

→ approximate this:

by just 1 training step supplementary

$\theta += \frac{\partial L(\hat{y}(x'))}{\partial \theta}$

$\frac{\partial L}{\partial \hat{y}(x')} \times \frac{\partial \hat{y}(x')}{\partial \theta}$

⇒ influence of point x' on point x:

$$\frac{\partial L}{\partial \hat{y}(x')} \cdot \frac{\partial L}{\partial \hat{y}(x)} = \frac{\partial \hat{y}(x')}{\partial \theta} \boxed{\frac{\partial L}{\partial \hat{y}(x')} \quad \frac{\partial L}{\partial \hat{y}(x)}} \frac{\partial \hat{y}(x)}{\partial (x)}$$

similarity kernel

↳ principled way for this

- would asking the network to change $\hat{y}(x)$ in a certain direction affect $\hat{y}(x')$ in the same direction?

↳ similarity score: $\frac{\partial \hat{y}(x')}{\partial x'} \cdot \frac{\partial \hat{y}(x)}{\partial x}$

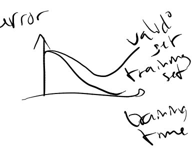
↳ Neural Tangent Kernel (NTK)

⇒ reliability ⟵ density / number of quite similar samples in the training set

- At the functional level

$x \to$ [B.B.] $\to \hat{y}$

F



during training: how is F evolving?

↓

project F on a 2D map for display

F? matrix $\hat{y}(x_i)$ for $x_i \in \mathcal{D}$

F_t: [matrix] ; $F_{t'}$: [matrix] — series of vectors → any dimensionality reduction technique → 2D

Information Bottleneck

Information theory

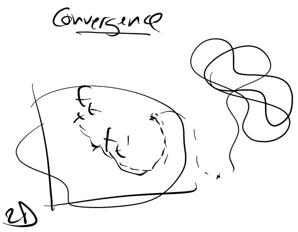
$x \to \square \to \square \to \square \to \square \to \hat{y}$

$\vec{A}_l(x)$

mutual information $I(X, \vec{A}_l(x))$ — layer l

$I(\vec{A}_l(x), \hat{Y})$

Convergence



General visualization tools

any kind of (high-dim) data ⟶ reduce dim ⟶ display
2D / 3D

can see $A(x)$ or $f_t(2D)$

↳ techniques : – PCA : linear (good for Gaussian-like data)
– non-linear : – t-SNE
– UMAP
– PyMDE Minimum Distortion Embedding

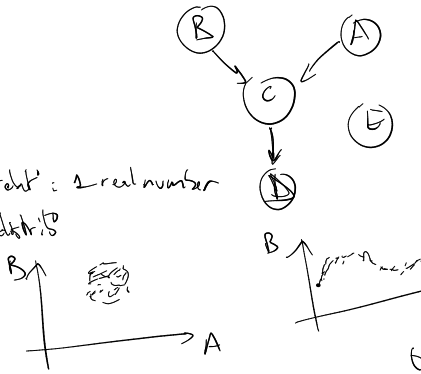
II Interpretability

– of web page

– causality : data : $\vec{x} = (A, B, C, D \dots)$
$\vec{x'} = (A', B', C', \dots)$

is A causing B or the opposite ?

↳ correlation : 1 real number
↳ full details



~~Correlation~~
causality

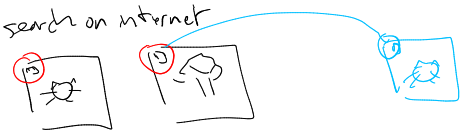
finding F
s.t. $B = f(A)$
is much simpler
than $A = f(B)$

– train NN
to estimate B from A
– " " A from B

III Issues related to datasets

Dataset poisoning

- dataset → search on internet



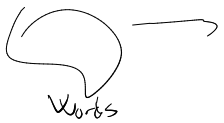
on purpose

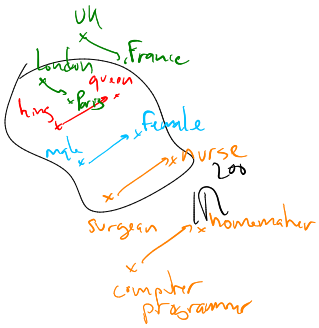
- biases

detecting cows ⇒ biased by background

Biases & fairness

- word2vec : tool mapping words ⟶ $\mathbb{R}^{200}$
↳ Google News
↳ trained for linear mappings :

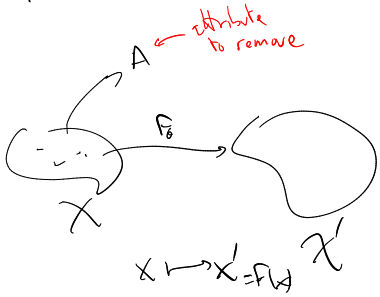




- check for fairness → def° ?

Solutions

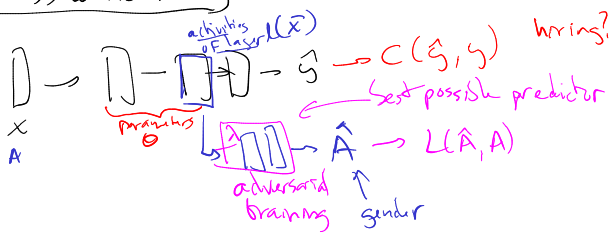
- preprocessing data :



Mutual information : $\max_{f_\theta} I(X, X') - I(A, X')$

- post-processing : trained network → adjust thresholds on outputs depending on A

- enforcing fairness while training:



θ: 2 objectives → good $\hat{y}$ + → increase $L(\hat{A})$

λ : 1 objective: decrease $L(\hat{A})$