Introduction to Information Theoretic Learning

June 24, 2015

Aim of this presentation

In this presentation we are going to discuss about *sparse filtering*.

Sparse filtering is a specific algorithm for unsupervised learning, but it provides the opportunity to discuss aspects of unsuperivsed learning that are general and conceptually stimulating.

Information

How do we quantify information?

We want a measure with (at least) the following two properties:

- **Extensivity**: given two mutually independent state A and B:

$$I(A \, and \, B) = I(A) + I(B)$$

- **Uncertainity Reduction**: if we are certain of a state A, then A does not carry any information

$$(P(A) = 1) \Rightarrow (I(A) = 0)$$

Hartley's Amount of Information

Let A be a generic event to which we can attribute probabilty $p(A)$.

Then the easiest function satisfying the requisites for being a measure of information is simply (negative) *logarithm*:

$$I(A) = -\log p(A)$$

Indeed:

$$I(A \, and \, B) = -\log p(A, B) \stackrel{(ind)}{=} -\log p(A)p(B) = I(A) + I(B)$$

and

$$I(A) = -\log p(A) = -\log 1 = 0$$

Information

From *general theory of means* we know that $H(\cdot)$[1] can take the following form:

We want a measure with (at least) the following two properties:

- **Extensivity**: given two mutually independent state A and B:

$$I(A\, and\, B) = I(A) + I(B)$$

- **Uncertainity Reduction**: if we are certain of a state A, then A does not carry any information

$$(P(A) = 1) \Rightarrow (I(A) = 0)$$

[1] We renamed here the more intuitive $I(\cdot)$ with the standard $H(\cdot)$

Unsupervised Learning (I)

Unsupervised Learning means learning from data $\mathcal{D}$ without external information.

Learn a good model generating *representation* of data

$$f : \mathcal{D} \to \mathcal{R}$$

Unsupervised learning is a underdetermined problem.

Unsupervised Learning (II)

What do we learn in absence of external guidance?

Two main approaches [**?**]:

- *Data Distribution Learning*: learn the *true* distribution of the data $\mathcal{D}$.
- *Feature Distribution Learning*: learn a *useful* distribution of the representations $\mathcal{R}$.

Data Distribution Learning

Data Distribution Learning is the traditional approach to unsupervised learning in which, given data $\mathcal{D}$, we try to model the distribution of the process that generated $\mathcal{D}$.

Several mainstream algorithms: *Boltzmann machines*, *autoencoders*, *indipendent component analysis* [**?**].

Implicit assumption: learning the *true structure of the data* (i.e.: the statistical description of the process generating the data) will automatically provide a *useful* representation.

Feature Distribution Learning (I)

Feature Distribution Learning is an innovative approach to unsupervised learning in which, given data $\mathcal{D}$, we try to model the distribution of the representation $\mathcal{R}$ in order to maximize its usefulness.

SF being the first algorithm of this kind [**?**].

Implicit assumption: some forms of representation are better than others and they will automatically provide a *useful* representation.

Feature Distribution Learning (II)

Assuming the conceptual framework of feature distribution learning we may now wonder:

1. *What sort of feature distribution may we want to learn?*
2. *How do we learn a feature distribution?*
3. *Is feature distribution learning feasible at all?*

Sparsity

1. What sort of feature distribution may we want to learn?

A *sparse* distribution, that is a distribution where most of the values are zero.

- *Practical reason:* sparse representation proved successful in many machine learning task (e.g.: *sparse deep belief networks* [**?**] or *k-sparse autoencoders* [**?**]);
- *Analogical reason:* biological systems implements sparse distributed representations (e.g.: modelling V1 cortex coding [**?**]);
- *Formal reason:* sparse distribution has low entropy[2] ([**?**])

[2] how important is this?

Sparse Filtering

2. How do we learn a feature distribution?

SF is an *algorithm* or *learning module* to perform *unsupervised feature distribution learning* that generates *sparse representations*.

Sparse Filtering

SF is an *algorithm* or *learning module* to perform *unsupervised feature distribution learning* that generates *sparse representations*.

Given a dataset[3]:

$$\text{raw features}\left\{\underbrace{\begin{bmatrix} .3 & .4 & .3 & \cdots & .7 \\ .2 & .7 & .3 & \cdots & .3 \\ .3 & .8 & .5 & \cdots & .6 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ .2 & .1 & .8 & \cdots & .4 \end{bmatrix}}_{\text{samples}}\right. \xrightarrow{SF} \left.\underbrace{\begin{bmatrix} 0 & 0 & 0 & \cdots & .7 \\ 0 & 0 & 0 & \cdots & .6 \\ 0 & .7 & 0 & \cdots & 0 \\ 0 & .8 & 0 & \cdots & 0 \\ .9 & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & .8 & \cdots & 0 \end{bmatrix}}_{\text{samples}}\right\}\text{SF features}$$

[3] notice the slightly unusual convention of having features along the rows and samples along the columns

SF - Sparsity

We achieve sparsity enforcing three properties:

1. *Population Sparsity:* each sample has few non-zero values;
2. *Lifetime Sparsity:* each feature has few non-zero values;
3. *High Dispersal:* activity on each row should be constant.

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & .7 \\ .7 & 0 & 0 & \cdots & .6 \\ 0 & .8 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & .8 & \cdots & 0 \end{bmatrix}$$

SF Algorithm

Minimize the following *loss function*

$$\operatorname{argmin}_W \left\| \left\| \left\| f\left(WX\right) \right\|_{L2,row} \right\|_{L2,column} \right\|_{L1}$$

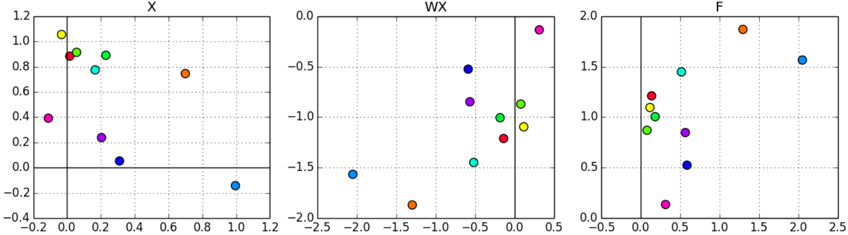
through *gradient descent*.

This ugly formula can be decomposed into four intuitive steps.

SF Algorithm - Step 1

Non-linear processing:

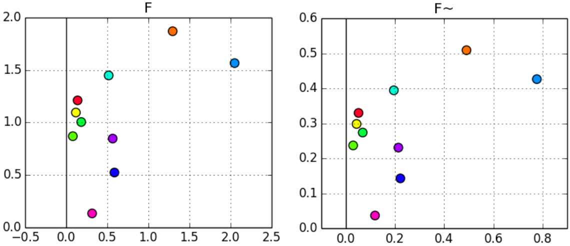
$$F = f(WX) = |WX|$$



SF Algorithm - Step 2

Normalization along the rows (features):

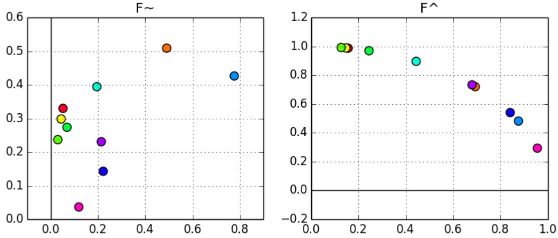
$$\tilde{F} = \frac{F}{\|F\|_{L2,row}}$$



SF Algorithm - Step 3

Normalization along the columns (samples):

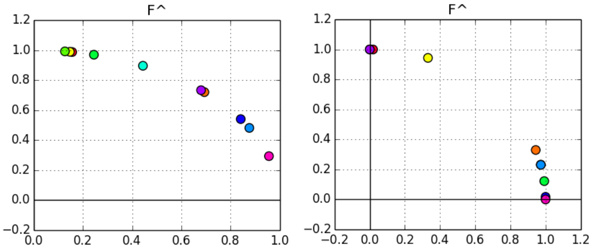
$$\hat{F} = \frac{\tilde{F}}{\left\|\tilde{F}\right\|_{L2,column}}$$



SF Algorithm - Step 4

Minimization of L1 norm:

$$\left\|\hat{F}\right\|_{L1}$$



SF Algorithm - Observations

1. *Population sparsity* is achieved by minimizing L1 norm;
2. *High dispersal* is enforced by row normalization by imposing the same mean square activation for each feature;
3. *Lifetime sparsity:* follows from the previous properties.

- These properties are imposed on the *feature distribution* not on the *data distribution*.
- Is the intution that we are performing a sort of *constrained scattering* correct?

Original Results

3. Is feature distribution learning feasible at all?

Results in [**?**]:

- *Timing and scaling:* SF shown to scale better than ICA, sparse coding or sparse autoencoders;
- *Processing of natural images:* SF learns Gabor-like filters;
- *Object classification on STL-10:* SF representations allows a linear SVM to achieve better performances than raw, random weights, k-means, and ICA representations;
- *Phoneme classification on TIMIT:* SF representations allows a linear or RBF SVM to achieve state-of-the-art performances.

Evaluating SF: Pros

- ✓ State-of-the-art performances
- ✓ Neat mathematical formulation
- ✓ Hyperparameter-light
- ✓ Highly computationally scalable
- ✓ Stackable
- ✓ Extensible[4]

[4] Can we use SF as a starting point to develop different feature distribution learning algorithms?

Evaluating SF: Cons

- × Data structure-agnostic
- × Fragility[5]
- × Sensitivity to initialization

[5] Could the extension of SF compromise the aim of learning a sparse distribution?

Discussion (I)

- *Is soft-absolute the best non-linearity?*
  [**?**] suggests the potential use of other functions, but no studies are available.
- *Is L2 normalization the best normalization?*
  It may be possible to project on other surfaces than a hypersphere, but no studies are available.
- *Can we earn something from stacking SF?*
  [**?**] run tests to study the filter-like behaviour of stacked SF, but no further results are reported.

Discussion (II)

- *When does the algorithm fail?*
  Testing shows that sometimes SF just fail in finding a sparse distribution.
- *How does failure relate to initialization?*
  Testing shows that radically different solutions are achieved with radically different initialization.
- *Can we prevent failure?*
  SF may be improved if the likelihood of failure could be decreased, or if failure-bound instances may be stopped earlier.
- *What is the optimal stopping criterion?*
  Literature show that the performances of SF strongly depends on the number of iterations.

Follow-ups: Practical Application

Thaler (1/218) [**?**] and *Romaszko* (6/218) [**?**] used SF successfully in the *Kaggle Black Box Challenge*.

Performance-oriented application of the SF algorithm in a classification pipeline.

Follow-ups: Practical Application

Ngiam et al.[6]	*Thaler*[7]
Research approach	Applicative approach
Shallow architecture (1 layer)	Deep architecture (2 layer)
Overcomplete representation	Undercomplete representation
Processing all the data in an unsupervised scheme	Training on training data and processing on testing data

[6] Some of these details are inferred from [**?**]

[7] These details are based on the technical report released by Thaler

Follow-ups: Comparisons and Extensions

- *Yang et al.* [?] extends the SF algorithm adding L2 weight regularization;
- *Zhang et al.* [?] compares different algorithms for unsupervised learning (including SF);
- *Lederer et al.* [?] suggests a connection between SF and *random matrix theory*;
- *Romero et al.* [?] improves on learning sparsity.

Some Further Questions

- *How to improve the sparsity learning algorithm?*
  Finding optimal ways to deal with failure and weight initialization.
- *How to apply sparse filtering in a train&test scenario?*
  Representation produced by SF depends on the other samples in the batch we are normalizing.
- *Can we use SF in a semi-supervised scenario?*
  SF may be used to learn representation out of big unlabelled datasets, before performing supervised or weakly supervised learning.
- *Can we use SF in a semi-superivised scenario where we aim at learning disentanglement?*

(Emotional) Disentangling Sparse Filtering

$$F = f(WX) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ where } \underbrace{\begin{matrix}[A] \\ [C]\end{matrix}}_{\substack{emo \\ sample}} \qquad \underbrace{\begin{matrix}[B]\} \\ [D]\}\end{matrix}}_{\substack{nemo \\ samples}} \begin{matrix} nemo\ features \\ emo\ \ features \end{matrix}$$

$$\mathcal{L} = \left\| \left\| \left\| \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_{L2,row} \right\|_{L2,column} \right\|_{L1} + \lambda_D \left\| D \right\|_{L1}$$

$$\mathcal{L} = \left\| \left\| \left\| \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_{L2,row} \right\|_{L2,column} \right\|_{L1} + \lambda_D \left\| D \right\|_{L1} + \lambda_A \left\| A \right\|_{L1}$$

Sparse Filtering Off-the-shelf

- *Matlab Implementation*:
  https://github.com/jngiam/sparseFiltering

- *Python Porting*:
  https://github.com/martinblom/py-sparse-filtering

References I