

Title: PSI 2018/2019 - Machine Learning - Lecture 11

Speakers: Roger Melko

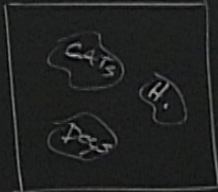
Collection: PSI 2018/2019 - Machine Learning (Hayward Sierens)

Date: April 08, 2019 - 9:00 AM

URL: <http://pirsa.org/19040006>

Unsupervised Learning (a Brief review)

U.L. Aims to model the underlying structure of

e.g. • Clustering.  → looking for inherent groups

• Association. -e.g. if you like X, you will also

I am Roger Melke.

structure of your data set. (data unlabeled)
inherent groupings.

you will also like Y (we will discuss with GM)

X_N) N bits

$\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m\}$

m is the size of your "training set"
or the number of measurements, etc.

Given a state (binary) $\vec{X} = (x_1, x_2, \dots, x_N)$ N

And the set of data $\mathcal{D} = \{\vec{X}\} = \{\vec{X}_1, \vec{X}_2, \dots$

$UL \neq RL \Rightarrow$ here we assume the data set $\mathcal{D}$ is given

$\mathcal{D}$ is distributed according to $P(\vec{X})$: can we discover

N bits

$\dots, \vec{x}_m\}$ m is the size of your "training set"
or the number of measurements, etc.
given to us for "free".
over an approximation $p(\vec{x}) \approx P(\vec{x})$ from the data above.
 $\uparrow$ approximation $\nwarrow$ unknown

$P(x)$: Can we describe the distribution of x in the state space? ↑ approximation

y distribution.

$$p(\vec{x}) \approx P_{data}(\vec{x}) = \frac{1}{|\mathcal{D}|} \sum_{\vec{x}_k \in \mathcal{D}} \delta_{\vec{x}, \vec{x}_k}$$

per than state space.

$\mathcal{D}$ is distributed according to $p(x)$. Can we...

Simplest way: Build the frequency distribution.

- Works if $||\mathcal{D}||$ is much larger than state space
- otherwise, it fails to generalize well.

Generative modelling: discover $p(x)$ that capture $P_{\text{data}}(\vec{x})$ but

↑ approximation

$$p(\vec{x}) \approx P_{\text{data}}(\vec{x}) = \frac{1}{|\mathcal{D}|} \sum_{\vec{x}_k \in \mathcal{D}} \delta_{\vec{x}, \vec{x}_k}$$

"memorizing
the
training set"



$p(\vec{x})$ but also generalizes well?

- 1) estimate $p(\vec{x})$ when $\vec{x}$ not in $\mathcal{D}$
- 2) make novel $\vec{x}$ that approximately fall under $P(\vec{x})$

The strategy is to model $P_{\text{data}}(\vec{x})$ with $p_{\lambda}(\vec{x})$

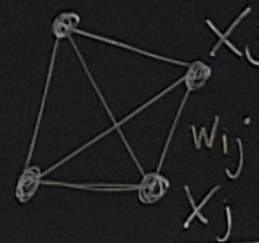
"Machine Learning" take $\mathcal{D}$, tune λ so that $p_{\lambda}(\vec{x})$

What model to use? Energy-based models (graphical re

Hopfield Network

$P_{\text{data}}(\vec{x})$ with $p_{\lambda}(\vec{x})$ with parameters λ . (of a neural
 , tune λ so that $p_{\lambda}(\vec{x}) \approx P_{\text{data}}(\vec{x})$ in a way that generalizes well
 based models (graphical representations of probability distributions).

$$E(\vec{x}) = - \sum_{i,j} W_{ij} x_i x_j - \sum_i b_i x_i$$



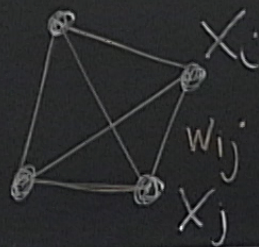
N nodes, $x_i =$

with $p_{\lambda}(\vec{x})$ with parameters λ . (weights & biases of a neural network)

so that $p_{\lambda}(\vec{x}) \approx P_{\text{data}}(\vec{x})$ in a way that generalizes well.

is (graphical representations of probability distributions).

$$\sum_{i,j} W_{ij} x_i x_j - \sum_i b_i x_i$$



N nodes, $x_i = 0, 1$

thus $p_i(\vec{x}) = \frac{1}{Z} e^{-E(\vec{x})}$, $Z = \sum_{\vec{x}} e^{-E(\vec{x})}$

ie. a fully-connected Ising model. (mag. field =

- used to model associative memory $\rightarrow$ local min are

a generalization: Boltzmann machine: add a hidden/

$e^{-E(\vec{x})}$ (partition function)

g. field = bias). Weight matrix ($J_{ij} = W_{ij}$) is fully symmetric.

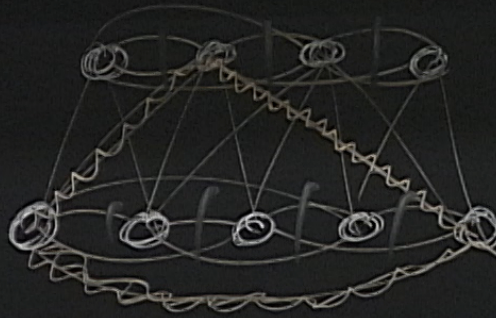
and are "memories" $\Rightarrow$ learning = optimization

$$P_i(\vec{x}) \approx P(\vec{x})$$

a hidden/latent space.

add a hidden/latent space.
- might be correlations difficult to model with Hopfield.

B.M.



- hidden ($W_h=4$)

- visible ($N=5$)

- might be correlated

- increase

o o o o

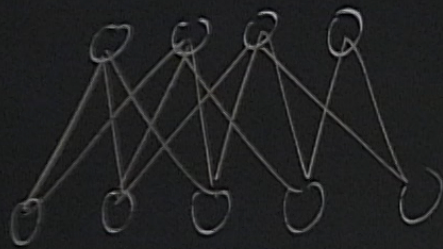
o o o o o

RBM: "Restricted Boltzmann machine"

— might be correlations difficult to model with Hopfield.

$N_n=4$) — increased representational capacity "expressiveness"

$N=5$)



hidden ← variable (a hyper parameter)
only intra-layer connections
visible ↑ fixed

$$b_i \sigma_i = \sum_{j=1}^n c_j h_j, \quad P_{\lambda}(\vec{V}, \vec{h}) = \frac{1}{Z_{\lambda}} e^{-E(\vec{V}, \vec{h})}$$

$$\lambda = (W, \vec{b}, \vec{c})$$

$\mathcal{D}$ is distributed according to $T(x)$: can we discover

define the energy: $E = E_{\lambda}(\vec{v}, \vec{h}) = -\sum_{ij} W_{ij} v_i h_j - \sum_{i=1}^m b_i v_i$

— Used in industry - efficient SGD learning algorithm.

• Generating unseen data (Recommender system)

discover an approximation for λ also.

$$-\sum_{i=1}^m b_i v_i - \sum_{j=1}^n c_j h_j, \quad P_{\lambda}(\vec{v}, h) = \frac{1}{Z_{\lambda}} e^{-E(\vec{v}, h)}$$

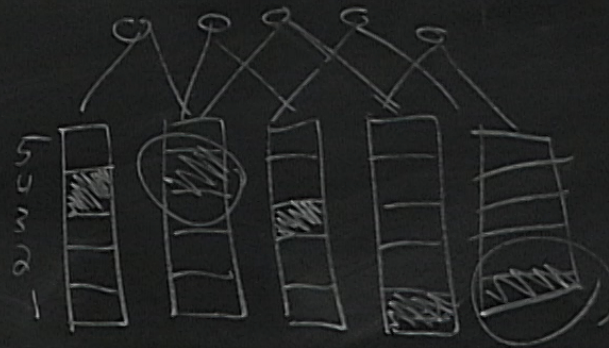
$\lambda = (W, \vec{b}, \vec{c})$

hm.
system) - Collaborative filtering (Hinton 2007)

the energy: $E = E_{\lambda}(\vec{v}, \vec{h}) = -\sum_{ij} W_{ij} v_i h_j -$

in industry - efficient SGD learning algorithm

• Generating unseen data (Recommender sys)





- Interpretation - (topic modelling - latent semantic structure)
 - generative pre training for deep classification

Back to $E(\hat{v}, \hat{h})$, $v_i = 0, 1$, $h_i = 0, 1$ $\Rightarrow$ Goal: train
 $\rightarrow$ After completion

Simplest way: Build the frequency distribution.

active)

classifiers

$\begin{matrix} n \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix}$

$\rightarrow$ FFNN

} pre-2012.

l: train λ on a set of data from $P(\vec{U})$

or complete sample $\vec{U}$ (and $\vec{h}$) $\Rightarrow$ goal of calculating
new physical observables

new physical observables

$\{\vec{V}_1, \dots, \vec{V}_M\}$ $\|\mathcal{D}\| = M$ drawn from an unknown $P(\vec{v})$

so that $P_\lambda(\vec{v}) = \sum_{\vec{h}} P_\lambda(\vec{v}, \vec{h}) \approx P_{\text{data}}(\vec{v})$

Tomorrow: Training . given $\mathcal{D} = \{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_M\}$

adjusting $\lambda = (W, \vec{b}, \vec{c})$ so that p_λ

Sampling (Generating) - generate new samples of $\vec{v}, h$ according

Estimators: $\langle \mathcal{O} \rangle_{p_\lambda(\vec{v}, h)} = \frac{1}{Z} \sum_{\vec{v}, h} \mathcal{O} p_\lambda(\vec{v}, h)$, e

new physical observables

$\vec{V}_2, \dots, \vec{V}_M$ $\|\mathcal{D}\| = M$ drawn from an unknown $P(\vec{v})$

so that $P_\lambda(\vec{v}) = \sum_{\vec{h}} P_\lambda(\vec{v}, \vec{h}) \approx P_{\text{data}}(\vec{v})$

$\vec{v}, \vec{h}$ according to $P_\lambda(\vec{v}, \vec{h})$

$P_\lambda(\vec{v}, \vec{h})$, e.g. $Q = E(\vec{v}, \vec{h})$

generalization. Boltzmann Machine - add a hidden layer.

But: define $O = O_v$ (act on visibles alone)

$$\langle O_v \rangle_{P_v(\vec{v}, \vec{h})} = \frac{1}{Z} \sum_{\vec{v}} O_v \underbrace{\sum_{\vec{h}} P_h(\vec{v}, \vec{h})}_{P_h(\vec{v}) \approx P(\vec{v})}$$

$$P(\vec{v})$$

$$Z = \sum_{\vec{v}} \sum_h P_h(\vec{v}, h)$$

$$= \sum_{\vec{v}} P_h(\vec{v})$$

$$p(h_i=1 | \vec{v}), p(h_i=0 | \vec{v}) = \sum_{\vec{h}} P_{\vec{h}}(\vec{v})$$

variable (a hyper parameter)

(n) hidden only intra-layer connections

(m) $\sqrt{3/64}$
 $\uparrow$ fixed