Budgeted Sequential Models for Data Processing

Ludovic Denoyer

May 4, 2017
Outline

1. Sequential Budgeted Learning
2. Budgeted Sequential Acquisition Models
3. **Focus on:** Deep Sequential Neural Networks [2014]
4. **Focus on:** Budgeted Reinforcement learning for Options Discovery [2017]
Introduction

Question

What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Question
What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Introduction

Question

What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Introduction

Question

What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Introduction

Question

What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Introduction

Question

What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Introduction

Question

What is the difference between how a human classifies these pictures and how a learning model (e.g. deep neural network) classifies these pictures?
Question

What is the difference between how a human classifies these pictures and how a learning model (e.g., deep neural network) classifies these pictures?
Budgeted Learning

ML Assumptions

- We assume that the input $x$ is *a priori* known (e.g., features are known)
  - *In the real life:* The information needs to be acquired
- The same computations are applied to all the inputs.
  - *In the real life:* Different computations are applied to different inputs
- The "size" of the model is chosen by hand
  - *In the real life:* The model is able to adapt itself to the context in which it is used

Research Direction from 2009 to now

- **Objective:** Develop a new family of models able to acquire information by themselves, to choose what to compute and to handle operational constraints.
Sequential Predictive Models

Inspiration: General Diagnostic Process

- Ask general questions to get context.
- Ask increasingly specific questions based on previous answers (Adaptive Acquisition)
- Exploit the answers in an adapted way (Conditional Computation)
- Build the output to predict (Output Prediction)

Key aspects:

- The model sequentially chooses actions to perform
- Actions can be of different types (acquisition, computation, prediction,..)
- The way actions are chosen is based on a cost-benefit trade-off
Sequential Budgeted Learning

Budgeted Sequential Learning

- **Input**: $x$ (a vector, a picture, a state/task, ...), **Output**: $y$ (a label, a vector, an action, ...)
- **Actions**: $H = H_1, ..., H_T$ a set of actions chosen following $P_\theta(H/x)$
- **Prediction**: A function $F_\theta(x, H) \rightarrow y$
- **Budget constraint**: $C(H)$ is the cost of $H$,
  - $C(H)$ can be the time spent for prediction, the memory/CPU/energy consumption, etc...
- **Learning objective**:

  $$J(\theta) = E_{P_\theta(x, H, y)}[\Delta(F_\theta(x, H), y) + \lambda C(H)]$$

  $$= \frac{1}{\ell} \sum_{k=1}^{\ell} E_{P_\theta(H/x)}[\Delta(F_\theta(x^k, H), y^k) + \lambda C(H)]$$  

(1)

Now, the learning problem also controls the behavior of the model, not only its predictive performance.
The origins: Structured Output Prediction

Firstly applied to structured output prediction

Actions are 'construction steps' (as in LaSo/Searn but with lighter assumptions), and $\Delta$ is a (non-differentiable) loss function (e.g. hamming loss) + No budget

Applications

- Sequence labeling (any order) [ECML/PKDD 2007]
- Node classification in Graph (with simulated inference) [ECML/PKDD 2009]
- Tree transformation problems [Machine Learning Journal 2009]

Learned with old and new approximated Reinforcement Learning algorithms (OLPOMDP, RCPI + Ranking).
Learning problem

Learning

\[ J(\theta) = \frac{1}{\ell} \sum_{k=1}^{\ell} \mathbb{E}_{P_{\theta}(H|x)}[\Delta(F_{\theta}(x^k, H), y^k) + \lambda C(H)] \] (2)

1. If \( J(\theta) \) is not differentiable \( \Rightarrow \) use your favorite Reinforcement Learning algorithm
2. If \( J(\theta) \) is differentiable \( \Rightarrow \) adapted gradient approaches

Sequential Acquisition Models [2010-now]

- an initial block of information \( x(0) \) is sampled
- the algorithm chooses which information \( x(t+1) \) to acquire next
- If the model considers that enough information has been gathered, it chooses to compute a prediction (a label here)
- The penalty is on the amount of information gathered or on its price (cost-sensitive classification)
Markov Decision Processes

- A state \( s \in S \) is composed of:
  - an input datum
  - a sequence of previously acquired information
- Different types of actions \( a \in A \):
  - Acquisitions actions
  - Classification actions
- Transitions: Can be stochastic or deterministic
- Reward: \(-\Delta(F_\theta(x, H), y) - \lambda C(H) (+ \text{ reward shaping})\)

Other elements

- Each state \( s \) is described by a features vector
  - It corresponds to the aggregation of previously acquired information
  - Can be manually defined or learned (e.g. recurrent neural networks [ICONIP 2016])
- A prediction is a policy: \( \pi(s) \rightarrow A \)
- An optimal policy is found by using approximating reinforcement learning algorithms
Examples of applications

Textual Reading process

- "The dry period means the temperaroi will be late this year. Again it seems that cocoa delivered earlier on consignment was included in the arrivals figures. In view of the lower quality over recent weeks farmers have sold a good part of their cocoa held on consignment. Comissaria Smith said spot bean prices rose to 340 to 350 cruzados per arroba of 15 kilos."
- "Classify as cocoa"

- (ECIR 2011) - Approximated RL with RCPI
- 4 text classification datasets
- Performance ≈ baseline methods (SVM)
- At a lower price...

Visual Attention Model

- (ICLR 2014) - specific learning Algorithm (close to Neural Fitted-Q)
- Results on 15-scenes and PPMI
Examples of applications

Features Acquisition

- Multiple MDPs structure studied and multiple reward functions
- (Machine Learning Journal 2012) - OLPOMDP and RCPI
- Recurrent Neural Network-based architecture
- Acquisition per block of features
- Deterministic and Stochastic models
- (ICONIP 2016), (IDA 2016)

<table>
<thead>
<tr>
<th>Task</th>
<th>Regularized Empirical Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard Budget</td>
<td>( \theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta(y^<em><em>\theta, y_i) + \lambda \frac{1}{2} \sum</em>{i=1}^{N} |z^</em>_\theta|<em>0 ) subject to ( |z^*</em>\theta|_0 \leq M ).</td>
</tr>
<tr>
<td>Cost-Sensitive</td>
<td>( \theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta_{cost}(y^<em><em>\theta, y_i) + \frac{1}{N} \sum</em>{i=1}^{N} \langle \xi, z^</em>_\theta \rangle )</td>
</tr>
<tr>
<td>Grouped Features</td>
<td>( \theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta(y^<em><em>\theta, y_i) + \lambda \frac{1}{N} \sum</em>{i=1}^{N} \sum_{\xi \in Z^</em><em>\theta} 1(f_i \subset Z^*</em>\theta) )</td>
</tr>
<tr>
<td>Relational Features</td>
<td>( \theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta(y^<em><em>\theta, y_i) + \frac{1}{N} \sum</em>{i=1}^{N} \sum_{f, f' \in Z^</em>_\theta} \text{Related}(f, f')(\lambda - \gamma) + \gamma )</td>
</tr>
</tbody>
</table>

Cold-start Recommendation

- Learning which questions to ask when a new user applies
- (ICLR 2014 – small paper) - Gradient descent
- Able to capture 'long-term' dependencies

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Nbitems</th>
<th>MF POP</th>
<th>MF HELF</th>
<th>IKNN POP</th>
<th>IKNN HELF</th>
<th>CS-IAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jester</td>
<td>5</td>
<td>0.603</td>
<td>0.589</td>
<td>0.604</td>
<td>0.634</td>
<td>0.667</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.613</td>
<td>0.609</td>
<td>0.640</td>
<td>0.608</td>
<td>0.686</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.665</td>
<td>0.641</td>
<td>0.688</td>
<td>0.676</td>
<td>0.701</td>
</tr>
<tr>
<td>MovieLens 1M</td>
<td>5</td>
<td>0.629</td>
<td>0.617</td>
<td>0.649</td>
<td>0.647</td>
<td>0.690</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.634</td>
<td>0.620</td>
<td>0.651</td>
<td>0.653</td>
<td>0.695</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.648</td>
<td>0.621</td>
<td>0.663</td>
<td>0.638</td>
<td>0.696</td>
</tr>
<tr>
<td>Yahoo</td>
<td>5</td>
<td>0.590</td>
<td>0.594</td>
<td>0.623</td>
<td>0.624</td>
<td>0.638</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.601</td>
<td>0.610</td>
<td>0.633</td>
<td>0.634</td>
<td>0.647</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.621</td>
<td>0.623</td>
<td>0.654</td>
<td>0.654</td>
<td>0.665</td>
</tr>
<tr>
<td>Flixtor</td>
<td>5</td>
<td>0.719</td>
<td>0.722</td>
<td>NA</td>
<td>NA</td>
<td>0.723</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.720</td>
<td>0.726</td>
<td>NA</td>
<td>NA</td>
<td>0.727</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.727</td>
<td>0.739</td>
<td>NA</td>
<td>NA</td>
<td>0.735</td>
</tr>
</tbody>
</table>
Conditional Computation

Given that the whole information as been acquired: Can we learn to compute good features for any input in an adaptive way?
Deep Sequential Neural Networks

Neural Networks as sequential data processing models

- A (Deep) neural network is a sequence of layers. At each time step:
  - An agent chooses to execute the "next" action
  - By executing this action, the input is transformed and transmitted to the next layer
- The transformations are learned from data

Question:
Deep Sequential Neural Networks extend the set of actions to multiple possible "routing" actions
A DSNN is a DAG with one input layer and one output layer. At each time step:

- An agent chooses which edge to execute (depending on the input)
- By executing this action, the input is transformed and transmitted to the next layer based on the chosen edge.
  - i.e each edge correspond to a (differentiable) data processing model.

- The transformations are learned from data
- The policy is learned simultaneously

Notations

- $H$ is a sequence of actions (edges)
- $F(x, H)$ is the prediction obtain on $x$ following $H$
- $f_{i,j}^\theta(\cdot) : \mathbb{R}^{N_i} \to \mathbb{R}^{N_j}$ is the $(i,j)$ layer function
  - $N_i$ is the size of the latent space for node $i$
- $P_i^\gamma(\cdot) : \mathbb{R}^{N_i} \to \mathbb{R}^{C_i}$ is the distribution probability over edges from $i$
  - $C_i$ is the number of children of node $i$
Deep Sequential Neural Networks

- A DSNN is a DAG with one input layer and one output layer. At each time step:
  - An agent (policy) chooses which edge to execute.
  - By executing this action, the input is transformed and transmitted to the next layer based on the chosen edge.

- The transformations are learned from data
- The policy is learned simultaneously

Algorithm 1 DSNN Inference Procedure

1: procedure INERENCE(x)
2: \( z^{(1)} \leftarrow x \)
3: \( n^{(1)} \leftarrow n_1 \)
4: \( t \leftarrow 1 \)
5: while not leaf \( n^{(t)} \) do  
6: \( a^{(t)} \sim p_{n^{(t)}}(z^{(t)}) \)  
7: \( n^{(t+1)} \leftarrow c_{n^{(t)},a^{(t)}} \)  
8: \( z^{(t+1)} \leftarrow f_{n^{(t)},n^{(t+1)}}(z^{(t)}) \)  
9: \( t \leftarrow t + 1 \)
10: end while
11: return \( z^{(t)} \)
12: end procedure
Deep Sequential Neural Networks

Learning problem

- Given a training set \( \{(x^k, y^k)\} \):
- Loss function:
  \[
  J(\theta, \gamma) = E_{P(x, H, y)}[\Delta(F(x, H), y)]
  \]
- Learning problem:
  \[
  \theta^*, \gamma^* = \arg \min_{\theta, \gamma} J(\theta, \gamma)
  \]

Gradient Computation

- Based on the REINFORCE trick but with a differentiable loss function instead of a reward

\[
\nabla_{\theta, \gamma} J(\theta, \gamma) = \int \nabla_{\theta, \gamma} (P(H|x)) \Delta(F(x, H), y) P(x, y) dH dx dy + \int P(H|x) \nabla_{\theta, \gamma} \Delta(F(x, H), y) P(x, y) dH dx dy
\]

\[
= \int \frac{P(H|x)}{P(H|x)} \nabla_{\theta, \gamma} (P(H|x)) \Delta(F(x, H), y) P(x, y) dH dx dy
\]

\[
+ \int P(H|x) \nabla_{\theta, \gamma} \Delta(F(x, H), y) P(x, y) dH dx dy
\]

\[
= \int P(H|x) \nabla_{\theta, \gamma} (\log P(H|x)) \Delta(F(x, H), y) P(x, y) dH dx dy
\]

\[
+ \int P(H|x) \nabla_{\theta, \gamma} \Delta(F(x, H), y) P(x, y) dH dx dy
\]

(3)
Deep Sequential Neural Networks

\[ \nabla_{\theta, \gamma} J(\theta, \gamma) = \frac{1}{\ell} \sum_{i=1}^{\ell} \left[ \frac{1}{M} \sum_{k=1}^{M} \nabla_{\theta, \gamma} (\log P(H| x_i)) \Delta(F(x_i, H), y) + \nabla_{\theta, \gamma} \Delta(F(x_i, H), y) \right] \]

Intuitive principle

- \( \nabla_{\theta, \gamma} (\log P(H| x_i)) \Delta(F(x_i, H), y) \) reinforces the path that has been chosen
- \( \nabla_{\theta, \gamma} \Delta(F(x_i, H), y) \) is the gradient of the selected neural network and can be computed by classical back-propagation
Deep Sequential Neural Networks

Tricks (that can be used or not)

- use a variance reduction term $\Delta(F(x_i, H), y) - b$ (e.g. computed over each minibatch)
- Add a regularization term for increasing exploration (entropy) during learning
- Add a regularization term for encouraging the use of all paths over mini-batches

Table 1: Examples of Decision Frontiers obtained on the Checkboard $7 \times 7$ dataset
Last topic: Budgeted Learning in Reinforcement Learning

- **Question:** Is the budgeted learning paradigm interesting for solving pure reinforcement learning problems?
Question

What is the difference between how a human solves this task and how a learning model (e.g., deep neural network trained with DQN/Actor-Critic/Policy gradient) solves this task?
Sequential Problems

Question

What is the difference between how a human solves this task and how a learning model (e.g., deep neural network trained with DQN/Actor-Critic/Policy gradient) solves this task?
Question

What is the difference between how a human solves this task and how a learning model (e.g. deep neural network trained with DQN/Actor-Critic/Policy gradient) solves this task?
Reinforcement Learning

- RL considers that observations are available at each time step
  - *In real life*: Observations are acquired by the agent (this is a choice)
- RL does not consider the price of information
  - *In real life*: Observation has a price (time to acquire information, time to process the information, ...)
- (Model-free) algorithms applies the same computations at each time step to choose an action
  - *In real life*: Sometimes you act quickly, sometimes, you have to think before acting.

Current Research: options discovery

- A task can be decomposed in sub-tasks (options = sub-policies)
- Question: How can we learn to discover such options?
- Answer: by using a budgeted learning approach
Options Discovery with Budgeted RL

Cognitive Effort

- **Assumption:** Options allows a **trade-off between performance and cognitive effort**.
- The cognitive effort is the price paid to solve the task e.g. time to think, energy consumption, ...
- **In this work:** Cognitive effort = Price of the observations

\[ N.B: \text{This idea is present in the neuro-science domain: habits/goal directed paradigm, model-free/model-based paradigm,...} \]

A budgeted RL context

- \( x_t \) is a low-level observation (at a low price). Can be "nothing" (blind setting)
- \( y_t \) is a high-level (expensive) observation.
The BONN Model

Budgeted Option Neural Network = BONN

BONN is a hierarchical neural network that automatically discovers options based on a budgeted learning objective (i.e. reduction of the cognitive effort)

Architecture

- The **actor model** computes the action (and keeps memory of the past)
- The **option model** computes a new option (and keeps memory of the past options)
- The **selection model** decides when an option needs to be computed
Learning BONN

Learning algorithm

- Budgeted reward:
  \[ r^*(s_t, H_t, \sigma_t) = r(s_t, H_t) - \lambda \sigma_t \]
  where \( \lambda \) is the additional price of high-level observations.

- Based on REINFORCE with two types of actions:

  \[
  \pi \leftarrow \pi - \gamma \sum_{t=0}^{T-1} \left( \nabla_{\pi} \log P(a_t|h_t) \right.
  + \nabla_{\pi} \log P(\sigma_t|h_{t-1}, x_t)) (R^*_t - b^*_t)
  \]

Three experimental settings

- **Blind setting**: \( x_t = 0, y_t = \text{observation (PO-MDP)} \)
  - e.g. in a maze: eyes closed / eyes opened.

- **Low/High level setting**: \( x_t = \text{local observation}, y_t = \text{global observation} \)
  - e.g. in a maze: what the agent sees / what god sees.

- **Instructional setting**: \( x_t = \text{local observation}, y_t = \text{optimal action} \)
  - e.g. in a maze: what the agent sees / "turn right", "go forward", ...
Blind setting

<table>
<thead>
<tr>
<th>Environment</th>
<th>R-PG</th>
<th>BONN $\lambda = 0.5$</th>
<th>BONN $\lambda = 1$</th>
<th>R-PG</th>
<th>BONN $\lambda = 0.5$</th>
<th>BONN $\lambda = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartpole</td>
<td>200</td>
<td>199.7</td>
<td>190.3</td>
<td>196.0</td>
<td>181.6</td>
<td>172.2</td>
</tr>
<tr>
<td></td>
<td>%obs</td>
<td>%obs</td>
<td>%obs</td>
<td>%obs</td>
<td>%obs</td>
<td>%obs</td>
</tr>
<tr>
<td>3 × 3-rooms</td>
<td>-3.3</td>
<td>-7.4</td>
<td></td>
<td>-14.9</td>
<td>-16.3</td>
<td></td>
</tr>
<tr>
<td>Lunar Lander</td>
<td>227.3</td>
<td>221.2</td>
<td>210.5</td>
<td>109.3</td>
<td>91.6</td>
<td>90.4</td>
</tr>
</tbody>
</table>

Conclusion

- The models can solve RL by using less observations...
- ...even in stochastic environments (i.e. $\epsilon = 0.25$)
Low/High level observations and Instructionnal setting

- $x_t =$ position in the room
- $y_t =$ shape of the room (goal if any + walls and doors)

- $x_t =$ 3 $\times$ 3 matrix around the agent
- $y_t =$ optimal action (computed by a planner)

Conclusion

- (left) The models is able to learn relevant options.
- (right) The models is able to learn when to ask for relevant instructions.
### Perspectives/Ongoing work

#### Online Budgeted Learning

Can we extend Sequential Budgeted Learning to consider a budget *during training*
- Current work on Meta Active Learning (each label has a price)

#### Model Free/Model Based

Can we naturally learn a model-free/model-based system in RL problems?
- It would allow the system spending time to think at key states.

#### Instructional RL

Can we learn a system able to ask questions when needed? Can we learn a system able to provide instructions when needed?
- It would allow the system to naturally interact with humans...
- ...and allows humans and computers to collaborate.
The end

Thank You