Learning What Data to Learn

Recent Trends in Automated Machine Learning (AutoML) (IN2107, IN4954)
Technische Universität München

Batuhan Erden
May 26th, 2021
An Adaptive Data Selection Strategy

- Curriculum Learning (CL), Bengio et al., 2009
- Self-paced Learning (SPL), Kumar et al., 2010

*How do we automatically and dynamically allocate appropriate training data at different stages of machine learning?*
An Adaptive Data Selection Strategy

Proposition: Two-fold intuitive principles!

The data selection strategy should be:
- General enough
- Forward-looking
The teacher-student framework

Figure credit: https://rlcurriculum.github.io/
Neural Data Filter (NDF)

- Deep Reinforcement Learning to determine whether/how to filter the given mini-batch of training data.
- The SGD training for the base Machine Learning model is casted into a Markov Decision Process (MDP).
The structure of SGD accompanied with NDF
NDF: Definition

SGD-MDP: \( < s, a, \mathcal{P}, r, \gamma > \)

- \( s \) is the state, \( s_t = (D_t, \mathcal{W}_t) \).
- \( a \) is the action space.
  - For data filtration task, we have: \( a = \{a_m\}_{m=1}^{M} \in \{0, 1\}^M \).
- \( \mathcal{P}_{ss'}^{a} = P(s'|s, a) \) is the state transition probability.
- \( r = r(s, a) \) is the reward.
- \( \gamma \in [0, 1] \) is the discount factor.
NDF: Definition

SGD-MDP: $< s, a, \mathcal{P}, r, \gamma >$

- NDF samples the action by its policy function $A = P_{\Theta}(a|s)$ with parameters $\Theta$ to be learnt.
- The policy $A$ can be any binary classification model, such as logistic regression.
The aim of designing state feature vector is to **effectively and efficiently** represent SGD-MDP state.

- Adopt 3 categories features to compose $f(s)$:
  - Data features
  - Base model features
  - Features to represent the combination of both data and model
Algorithm 1: SGD Training with NDF

**Algorithm 1** SGD Training with Neural Data Filter.

**Input:** Training data $D$.

1. Randomly sample a subset of NDF training data $D'$ from $D$.
2. Optimize NDF policy network $A(s; \Theta)$ based on $D'$ by policy gradient (details in Algorithm 2).
3. Apply $A(s; \Theta)$ to full dataset $D$ to train the base machine learning model by SGD.

**Output:** The base machine learning model.
We aim to optimize the following expected reward:

\[ J(\Theta) = E_{P_\Theta(a|s)}[R(s, a)] \]

\[ \nabla_{\Theta} = \sum_{t=1}^{T} E_{P_\Theta(a_1:T|s)}[\nabla_{\Theta} \log P(a_t|s_t)R(s_t, a_t)] \]

Which is empirically estimated as:

\[ \sum_{t=1}^{T} \nabla_{\Theta} \log P(a_t|s_t)v_t. \]
Algorithm 2: Train NDF policy

**Algorithm 2** Train NDF policy.

**Input:** Training data $D'$. Episode number $L$. Mini-batch size $M$. Discount factor $\gamma \in [0, 1]$.
- Randomly split $D'$ into two disjoint subsets: $D'_{train}$ and $D'_{dev}$.
- Initialize NDF data filtration policy $A(s, a; \Theta)$, i.e., $P_{\Theta}(a|s)$.

**Output:** The NDF policy $A(s, a; \Theta)$.

**for** each episode $l = 1, 2, \cdots, L$ **do**

Initialize the base machine learning model.
Shuffle $D'_{train}$ to get the mini-batches sequence \{D_1, D_2, \cdots\}.
$T = 0$.

**while** stopping criteria is not met **do**

$T = T + 1$.
Sample data filtration action for each data instance in $D_T = \{d_1, \cdots, d_M\}$: $a = \{a_m\}_{m=1}^M$, $a_m \sim P_{\Theta}(a|s_m)$, $s_m$ is the state corresponding to $d_m$.
Update base machine learning model by Gradient Descent based on the selected data in $D_T$.
Receive reward $r_T$ computed on $D'_{dev}$.

**end while**

**for** $t = 1, \cdots, T$ **do**

Compute cumulative reward $v_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t} r_T$.

$$\Theta \leftarrow \Theta + \alpha v_t \sum_m \frac{\partial \log P_{\Theta}(a_m|s_m)}{\partial \Theta}$$  

(5)

**end for**

**end for**
The experiments

- Unfiltered SGD
- Self-paced Learning (SPL)
- NDF

\[ \Theta \leftarrow \Theta + \alpha (r_t - b_t) \sum_m \frac{\partial \log P(a|s_m)}{\partial \Theta} \]

, with the reward baseline \( b_l \) for episode \( l \), computed as \( b_l = 0.8b_{l-1} + 0.2r_l, b_0 = 0 \).

and \( v_t \) is computed as \( v_t = r_l \).

- RandDrop
MLP for MNIST

- 60k training and 10k testing images
- **Optimizer**: Momentum-SGD (Mini-batch size = 20).
- **Layer structure**: A 3-layer feedforward neural network with layer size 784 x 500 x 10
- **Loss function**: Cross-entropy
- **L**: 500 Episodes
- **Early Stopping**: Based on validation set accuracy
MLP for MNIST
MLP for MNIST
CNN for Cifar10

- 60k RGB images of size 32 x 32 categorized into 10 classes.
- 50k training images and 10k test images.
- **Data Augmentation:** Padding 4 pixels to each side and randomly sampling a 32 x 32 crop.
- **ResNet** is adopted.
- **Optimizer:** Momentum-SGD (Mini-batch size = 128).
- **Learning Rate:** Initially set to 0.1, and multiplied by a factor of 0.1 after the 32k-th and 48k-th model update.
- **L:** 100.
CNN for Cifar10
CNN for Cifar10

![Graph showing filter number against epoch for different buckets. The legend includes Bucket 1 to 26, Bucket 27 to 52, Bucket 53 to 77, Bucket 78 to 103, and Bucket 104 to 128. The graph illustrates the trend of increasing filter number with the number of epochs.]
RNN for IMDB sentiment classification

- 50k movie review comments with positive/negative sentiment labels
- 25k training set and 25k test set.
- The size of word embedding in RNN is 256.
- The size of hidden state of RNN is 512.
- Mini-batch size is 16.
- \( L: 200 \).
RNN for IMDB sentiment classification
RNN for IMDB sentiment classification
Pros

• A good data selection mechanism can effectively accelerate model convergence.
• Different tasks and datasets may favor different data selection policies.
• Not sensitive to the setting of hyper-parameters.
Cons

- For some problems, NDF seems to be indifferent to different setting of hyper-parameters.
- The data selection (hard or easy) matters in some problems.
- Still need to test the algorithm on different scenarios.
Questions???