How to Squeeze the Training Data Efficiently and Effectively?

November 24, 2015

1 Introduction

Training samples are not equally valuable [4]. The next natural question is how can we squeeze the training data efficiently and effectivelly? Self-paced learning [3] has a similar flavor to active learning, which chooses a sample to learn from at each iteration, though the labels of all the samples are not known when the samples are chosen in the active learning setting. Active learning approaches differ in their sample selection criteria. For example, it is suggested in [9] to choose a *hard* sample which is close to the margin. In [1] the most *uncertain* sample is preferred with respect to the current classifier. In contrast, self-paced learning advocates focusing on the *easier* instances first and progressively expanding its repertoire to include more complex ones [7, 5, 3].

These lead to a question of whether for different data sets and different learning algorithms, the ordering of training samples might actually be diverse. In other words, true *self-paced* learning should have adaptive orderings, maybe from easy to hard or from hard to easy, which depends on the specific data sets and learning algorithms.

Different from all the previous self-paced learning methods based on a manually-designed and domain-specific easiness functions [7, 5, 3], we propose a generic and truly *self-paced* learner based on a Deep Q-Network (DQN), which does not assume an artificial ordering of training samples in advance.

The individual techniques of the proposed self-paced learning we outline are not completely novel, but we believe this generic self-paced learning deserves to be more widely known and practiced due to its compatibility.

We also consider the optimal ratio of positive and negative training samples.

2 Contribution

Propose a truly generic self-paced learning framework, which can work with *all* existing classifiers without changing their implementations. Our method only requires a limited "black box" interface with the models, allowing us to use very sophisticated, state-of-the-art classifiers without having to look under the hood.

3 Related Work

The stochastic version of Bayesian optimization for big data tasks is first introduced in [6]. More concretely, those evaluations on subsets are simply treated as evaluations of a latent performance curve corrupted by noise.

The structure of iterative learning algorithms is first considered in [8], where the training loss during the fitting procedure is assumed to roughly follows an exponential decay towards an unknown final value. A Bayesian non-parametric prior around this assumption is built by developing a new kernel that is an infinite mixture of exponentially-decaying basis functions, with the goal of characterizing these training curves.

4 Setup

In this work, the stochastic setting is assumed. Multiple batches $S_t, t = 1, ..., T$ are given, and each S_t consists of pairs of examples $\{x_i, y_i\}$, where the learner is to predict the label $y_i \in \mathcal{Y}$ of the instance $x_i \in \mathcal{X}$. It is assumed that the data are generated independent and identically from an underlying data-generating distribution $\mathcal{D} = \{\mathcal{X} \times \mathcal{Y}\}$. The goal of our proposed Bayesian optimizer is to find a sequence of training samples (multiple passes over the whole data set are allowed) for a learning algorithm, such that the training loss will roughly follow a *smooth* exponential decay towards an unknown final value as quickly as possible while with a constraint on the decreasing rate.

5 Procedure

Initially, a learning algorithm is trained with a random subset of training samples. Then all the training samples are sorted by the confidence scores given by the partially trained learning algorithm.

The subsequent selections of the next mini-batches of samples are modeled as a Gaussian process bandit problem, where each arm is specified by a normalized confidence score and the reward is the associated training error. For example, an arm might be "confidence score = 0.3", and then the training samples around that score will be chosen to form a mini-batch.

One key step is to put a constraint on the decrease in training error with the help of [2]. Intuitively, the decrease in training error should not change dramatically. Suppose the sequence of decreases in training error is $[tr_1, ..., tr_T]$, then $\frac{tr_t}{tr_{t-1}} \approx C$, where C is an unknown constant.

6 Experiments

We would like to see whether the selection of training samples makes the training algorithms converge and to determine in which way it could converge to a better final solution.

References

[1] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *arXiv preprint cs/9603104*, 1996.

[2] Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*, 2014.

[3] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[4] Agata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*, 2013.

[5] Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1721–1728. IEEE, 2011.

[6] Thomas Nickson, Michael A Osborne, Steven Reece, and Stephen J Roberts. Automated machine learning on big data using stochastic algorithm tuning. *arXiv preprint arXiv:1407.7969*, 2014.

[7] James Steven Supancic III and Deva Ramanan. Self-paced learning for long-term tracking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2379–2386. IEEE, 2013.

[8] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[9] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.