Small Coresets Construction for Machine Learning

Zengfeng Huang

Fudan University

1 Background

We are given a very large labeled training data set $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, where y_i is the corresponding label of x_i. For a machine learning model $\mathcal{M}$, let Θ denote its learnable parameters. We want to find the optimal Θ that minimizes the empirical loss of $\mathcal{M}$ with respect to $\mathcal{D}$, which is defined as

$$L(\Theta) = \sum_{(x_i,y_i)\in\mathcal{D}} \ell(\mathcal{M}(x_i, \Theta), y_i),$$

where $\ell(\cdot,\cdot)$ is a loss function defined on each example, e.g. l_2 loss, cross-entropy loss, etc. Let Θ^* be the optimal parameters for $\mathcal{M}$, i.e.

$$\Theta^* = \arg\min_{\Theta} L(\Theta).$$

In our application setting, we have a set of m different candidate models $\{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$ and wish to decide which model is the best for the learning problem at hand. Naively, we can train each model on the training set $\mathcal{D}$, evaluate their validation accuracy, and select the one with highest validation accuracy. The challenge is that the models may have millions of parameters each and the training set is huge. Thus, it could takes days to train each model and weeks to train all the candidate models. Our goal is to design efficient methods to select the (approximately) best model without fully training all models on the entire training set.

Our idea is to train each model only on a small subsample carefully chosen from $\mathcal{D}$ (the subsample could potentially be different for each candidate model), and select the model with the best testing accuracy. To make this idea work, roughly speaking, we want the "rank" of the candidate models to be preserved when trained with the small set. The main challenge is how to choose this subsample, since in general, a model performs well with a small training set doesn't necessarily perform better when the full training set is used.

1.1 Problem Formulation: ε-Coreset

Let $\mathcal{D}' \subset \mathcal{D}$ a subset of training samples, where each $x_i \in \mathcal{D}'$ is assigned with a positive weight $w_i > 0$. Define the weighted empirical loss of $\mathcal{M}$ with respect to $\mathcal{D}'$ as

$$L'(\Theta) = \sum_{(x_i,y_i)\in\mathcal{D}'} w_i \ell(\mathcal{M}(x_i, \Theta), y_i).$$

We wish $L'(\Theta) \approx L(\Theta)$ to hold for all Θ simultaneously. Intuitively, if this is the case, we only need to train $\mathcal{M}$ on $\mathcal{D}'$ with loss function L'.

An ε-Coreset is a weighted subset $\mathcal{D}' \subset \mathcal{D}$ with each $x_i \in \mathcal{D}'$ assigned a positive weight $w_i > 0$, such that for all Θ, we have

$$(1-\varepsilon)L(\Theta) \leq L'(\Theta) \leq (1+\varepsilon)L(\Theta).$$

For a given training set $\mathcal{D}$ and model $\mathcal{M}$, our goal is to find an ε-coreset of minimum size.

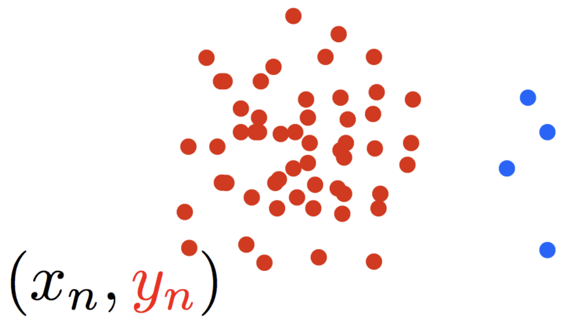
2 Methods

2.1 Importance sampling

We independently sample each point $x_i \in \mathcal{D}$ with probability p_i; if it is sampled we assign a weight $w_i = \frac{1}{p_i}$ to x_i and if it is not sampled, $w_i = 0$. It is easy to see the expectation of w_i is $\mathsf{E}[w_i] = 1$, and also

$$\mathsf{E}[L'(\Theta)] = \mathsf{E}[\sum_{(x_i,y_i)\in\mathcal{D}} w_i \ell(\mathcal{M}(x_i,\Theta),y_i)] = \sum_{(x_i,y_i)\in\mathcal{D}} \mathsf{E}[w_i]\ell(\mathcal{M}(x_i,\Theta),y_i) = L(\Theta).$$

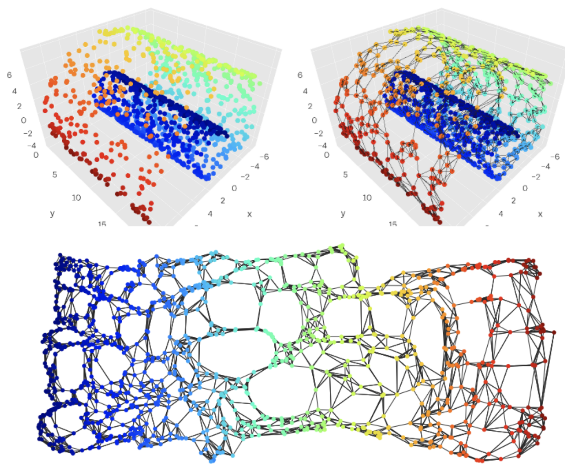
Therefore, we at least have $L'(\Theta) = L(\Theta)$ in expectation for all Θ. However, this is not enough, we still need to worry about the variance. The variance is highly correlated with the sampling probabilities $p_1, \cdots, p_n$ and uniform sampling (i.e., all p_i's are equal) is usually not good even for simple binary linear classification. For example, if the two classes are extremely unbalanced, uniform sampling would result in a subset with examples all from one class (the red class in the following figure).



Intuitively, the blue points in the above figure is more “important” than the red points, and should be sampled with higher probabilities. In fact, the key of the importance sampling method is how to characterize the “importance” of each training example, and is often highly non-trivial.

2.2 Importance Scores Based on Structural Information

We propose to characterize the importance of each point by looking at its structural information. It is widely assumed that (high-dimensional) data points in the training set are approximately from a low-dimensional manifold. To explore this property, some type of graphs are constructed whose nodes are the data points, e.g. kNN graphs, which can be viewed as a discretization of the manifold.



Then, one can apply graph-based method (e.g., spectral clustering) for solving learning tasks. For our purpose, after the graph is constructed, we characterize the importance of each point, i.e., each node on the graph, using graph-theoretical tools, e.g., pagerank, random walk, centrality, diffusion, etc. Moreover, we can apply *Graph Neural Networks* to automatically learn the importance of each node.

2.3 Model-dependent Importance Scores

To learn the importance, the above method only consider data distribution information. However, the importance of the same point in the data set could be very different for different models. Thus, it could be of great help to utilize the properties of the model being trained to design model-dependent importance scores. For example, we could use Influence Functions, a classic tool from statistics, to characterize the importance of each point with respect to the model. It is also helpful to use a small proxy model to perform data selection, e.g., by removing hidden layers from the target model, using smaller architectures, and training for fewer epochs. Although these small proxy models have higher error rates, they could provide useful signals for coreset construction.

2.4 Active Data Selection

In the above methods, importance scores of all points and the subsample $\mathcal{D}'$ are determined in the beginning. Given $\mathcal{D}'$, we then train the model with stochastic mini-batch gradient methods. However, as the training process goes on, the inherent importance of each point may change. So, it is natural to assign importance scores in an adaptive manner. More precisely, we may sample the next mini-batch of data based on the current importance scores. For this purpose, we can explore ideas from *active learning*. Conventional active learning selects points to label from a large pool of unlabeled data by repeatedly training a model on a small pool of labeled data and selecting additional examples to label based on the model's uncertainty (e.g., the entropy of predicted class probabilities) or other heuristics.

3 Plan

Since model-dependent methods are more complicated and it is much harder to make them work in practice, we mainly focus on model independent methods and the use of graph techniques. Coresets are widely studied for simpler problems such as k-means, logistic regression, Bayesian inference etc., however, there is currently not much related work on general deep learning models. So, in the next stage, we need to implement and test some baseline methods for common data sets, eg. MINIST, CIFAR10, CIFAR100, ImageNet, and then extend these baseline to more sophisticated and effective approaches. The following two main challenges needs to be addressed: 1) how to exploit the properties the data manifold and what is the right way to build the underlying graph; 2) once the graph is built, which methods should be used to learn the importance scores for each node/example. We've been doing research on graph-based learning methods in recent years, and believe such techniques will be of great help for our tasks. Next, we'll move to more realistic data sets and applications from Tencent. Finally, we will explore model-dependent and active learning methods.