Stochastic Gradient Descent For Budgeted Learning

August 6, 2015

1 Introduction

In supervised classification involving a large number of features, the computation of the features should be taken into account. Indeed, if time or cost matters, it not always beneficial to compute all the features before making a decision. For example, in image categorization, a simple classifier based on a thumbnail images might be good most of the time, but if the object inside the image is small or occluded, one might require to download the full resolution image in order to make the final decision about the category. Similarly, when focusing on applications such as medical diagnostic, simple measures can be made (e.g. the color of the skin, the presence of cough, temperature, etc.) but they might not be discriminative, while more complex but costly diagnostics might be needed to have further information to predict the presence or absence of a disease.

The notion of cascading features is the fact that a set of features is computed dynamically, based on the information we have gathered so far.

Several approaches have been proposed by casting the problem as a reinforcement learning problem [Citations to be made].

Unlike previous approaches based on reinforcement learning, we treat the problem directly by

feng: Comments of Feng appear here

venkat: Comments of Venkat appear here

joe: Comments of Joe appear here

2 Decision-theoretic framework

For simplicity we first introduce our approach in a simple setting when we only have two types of features, and these features are observed all together. Consider the regression task where the goal is to predict an output $y \in \mathcal{Y}$ based on "cheap" features $x_1 \in \Re^{d_1}$ and "costly" features $x_2 \in \Re^{d_2}$.

We assume that for every prediction s, the user can choose whether or not computing x_2. This binary choice, denoted δ is based on a rational decision of

the agent (the user) that tries to minimize its expected loss. If $\delta = 1$, only x_1 is observed and we incur a loss ℓ for predicting $s_1 \in \Re$ based only on on x_1. If $\delta = 2$, the user decides to observe x_2. In such case, the total loss $\bar{\ell}$ is then a fixed amount c_2 plus the loss ℓ of predicting $s_2 \in \Re$ that has been made based on x_1 jointly with x_2.

$$\bar{\ell}(x_1, x_2; \delta, y^*) = \ell(s_1(x_1, \theta_1); y^*)\mathrm{I}_{\{\delta=1\}} + (c_2 + \ell(s_2(x_1, x_2, \theta_2); y^*))\,\mathrm{I}_{\{\delta=2\}} \quad (1)$$

where y^* is the ground truth output, $s_1(x_1, \theta_1)$ is the model for the cheap features and $s_2(x_1, x_2, \theta_2)$ is the model for the expensive features, $\theta_1 \in \Theta_1$ and $\theta_2 \in \Theta_2$ being parameters of the two models. Estimators for θ_1 and θ_2 can be obtained by minimizing the empirical loss on the training data, possibly penalized by a regularization function to avoid overfitting:

$$\hat{\theta}_1 \in \arg\min_{\theta_1 \in \Theta_1} \sum_{i=1}^{n} \ell(s(x_1^{(i)}, \theta_1); y^{(i)}) + \lambda_1 \Omega_1(\theta_1) \quad (2)$$

$$\hat{\theta}_2 \in \arg\min_{\theta_2 \in \Theta_2} \sum_{i=1}^{n} \ell(s(x_1^{(i)}, x_2^{(i)}, \theta_2); y^{(i)}) + \lambda_2 \Omega_2(\theta_2) \quad (3)$$

where the training dataset is represented by n cheap input-expensive input-output triples $\{x_1^{(i)}, x_2^{(i)}, y^{(i)})\}$.

Given x_1, the oracle decision δ^* is obtained by minimizing the expected loss $\min_\delta L^* = \mathrm{E}_*[\ell(x_1, x_2; \delta, y^*)|x_1]$ with respect to δ under the sample distribution P^*. This means that the best possible decision would be:

$$\begin{aligned} \delta^*(x_1) &= 1 \qquad \text{if } \mathrm{E}_*[\ell(s_1(x_1); y^*)|x_1] < c_2 + \mathrm{E}_*[\ell(s_2(x_1, x_2); y^*)|x_1] \\ &= 2 \qquad \text{otherwise.} \end{aligned} \quad (4)$$

We do not know P^*, but we can estimate this decision function on the training data. It is a binary decision problem, so learning this decision function $a(x_1; \pi)$ is equivalent to a regression that uses x_1 as input features, where $\pi \in \Pi$ are the parameters to be estimated. The binary output variables associated to every training instance are denoted by $\delta^{(i)}$, $i = 1, \cdots, n$ and are obtained by using the decision rule of Equation(4) but using empirical expectations rather than exact expectations.

$$\hat{\pi} \in \arg\min_{\pi \in \Pi} \sum_{i=1}^{n} \ell_\sigma(a(x_1^{(i)}, \pi); \delta^{(i)}) + \lambda_0 \Omega_0(\pi) \ , \quad (5)$$

where ℓ_σ is the logistic loss:

$$\begin{aligned} \ell_\sigma(u, \delta) &= \log(1 + e^{-u}) \qquad \text{if } \delta = 1 \\ &= \log(1 + e^{u}) \qquad \text{otherwise.} \end{aligned} \quad (6)$$

but an alternative objective that should give similar performances is based directly on the scores:

$$\begin{aligned}\hat{\pi} \quad \in \quad & \arg\min_{\pi \in \Pi} \sum_{i=1}^{n} \sigma(a(x_1^{(i)}, \pi)) \ell(s_1(x_1^{(i)}, \hat{\theta}_1); y^{(i)}) \\ & + \sigma(-a(x_1^{(i)}, \pi))(c_2 + \ell(s_2(x_1^{(i)}, x_2^{(i)}, \hat{\theta}_2); y^{(i)})) + \lambda_0 \Omega_0(\pi) \ . \end{aligned} \tag{7}$$

Here, we directly see that if the decision function $a(x_1^{(i)}, \pi)$ is able to overfit on the training data, the empirical loss minimized by Equation (7) is the smallest training loss for this combination of parameters $\hat{\theta}_1$ and $\hat{\theta}_2$.

3 Budgeted Learning

The previous approach works fine to learn the optimal decision at test time. But the implicit assumption we made is that we are in a batch setting, where the training data have been already fully observed, but this can cost a lot for problems with a large number of training data. More precisely, the cost of acquisition during training is $n \times c_2$, where n is the total number of training data. There are two issues with the previous approach:

- Learning θ_2 requires expensive features.
- Learning the optimal decision function δ requires expensive features to be extracted from training data.

Ideally, we would like to be able to learn a *good enough* decision function by observing the expensive features for only a subset n_2 of the training data where $n_2 << n$.

Finding the optimal strategy to acquire training data is an instance of active learning scenario. A every acquisition step of expensive features, we can use the current estimate of the parameters to prediction which training instances are more likely to be better estimated using the expensive features. A naive approach is to use the model uncertainty based only on the cheap features, the intuition being that the interesting cases are the ones that cannot be well predicted using the cheap features. This approach will serve as baseline

Guillaume: IMPORTANT POINT TO BE DISCUSSED: Do we want to compare with standard active learning?

In the previous section, we *jointly minimize the loss and the decision function parameters during training.* This means that we solve the following unrelated problems:

Instead, we propose to cast Equations (2), (3) and (7) into one single optimization problem.

$$(\hat{\theta}_1, \hat{\theta}_2, \hat{\pi}) \in \arg \min_{\theta_1 \in \Theta_1, \theta_2 \in \Theta_2, \pi \in \Pi} \sum_{i=1}^{n} \sum_{\delta=1}^{2} P_\pi(\delta | x_1^{(i)}) \ell_{i\delta}(\theta_1, \theta_2) + \lambda_0 \Omega_0(\pi) + \lambda_1 \Omega_1(\theta_1) + \lambda_2 \Omega_2(\theta_2) \quad (8)$$

where:

$$P_\pi(\delta = 1 | x_1^{(i)}) = \sigma(a(x_1^{(i)}, \pi)) \quad (9)$$

$$\ell_{i1}(\theta_1, \theta_2) = \ell(s_1(x_1^{(i)}, \theta_1); y^{(i)}) \quad (10)$$

$$\ell_{i2}(\theta_1, \theta_2) = c_2 + \ell(s_2(x_1^{(i)}, x_2^{(i)}, \theta_2); y^{(i)}) \quad (11)$$

and we minimize (8) using SGD:

At each iteration until convergence:

- Sample i and observe the cheap feature $x_1^{(i)}$
- Sample δ using $P(\delta | x_1^{(i)})$
- If $\delta = 1$, then
  - $\theta_1 \leftarrow \theta_1 - \epsilon_1 \nabla_{\theta_1} \ell(s_1(x_1^{(i)}, \theta_1); y^{(i)})$
- Else ($\delta = 2$)
  - Sample i and acquire the expensive feature $x_2^{(i)}$
  - $\theta_2 \leftarrow \theta_2 - \epsilon_2 \nabla_{\theta_2} \ell(s_1(x_1^{(i)}, x_2^{(i)}, \theta_2); y^{(i)})$
  - $\pi \leftarrow \pi - \epsilon_0 P_\pi(\delta | x_1^{(i)}) \ell_{i\delta}(\theta_1, \theta_2) \nabla_\pi \log P_\pi(\delta | x_1^{(i)})$

In this algorithm, we see that we sample the decision δ based on our current model P_π. We can verify that the gradient is correct and converges to the same solution as in the batch setting, but we do not always make the acquisition of the expensive feature x_2. This algorithm can be seens as a *on-policy* reinforcement learning algorithm, and can be seen as a special instance of the Reinforce algorithm [cite Ronald J. William, 1987].

Further improvements can be made by using off-policy learning algorithms.