A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data

Presenter: Zhe Wang

https://qdata.github.io/deep2Read

Zhe Wang

201909

Background

For supervised learning tasks:

The training set $\{x_i, y_i\}$ is i.i.d sampled from the unknown distribution $D = P(X, Y)$. The hypothesis set $\mathcal{H}$ contains a family of mapping from $X \rightarrow Y$.

Find the optimal mapping in $\mathcal{H}$ by expected risk minimization:

$$f^* = \arg\min_{f \in \mathcal{H}} E_{(x,y)\sim D} L(f(x), y)$$

Due to the unknown D, empirical risk minimization will be minimized instead.

$$\hat{f}^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{N} L(f(x_i), y_i)$$

The distance between the $\hat{f}^*$ and the true but unknown mapping g can be factorized into two terms:

- The distance between $\hat{f}^*$ and f^* is called estimation error.
- The distance between g and the hypothesis set $\mathcal{H}$ is known as the approximation error.

For supervised learning, the trade off between estimation error and approximation error is searched by cross validation.

Learning a good hypothesis space

A good hypothesis set should have a small estimation error and approximation error.

Basic assumption: If one observes multiple prediction problems, then we can better estimate the underlying hypothesis space.

- m learning problems indexed by $l \in \{1, \cdots, m\}$,
- For problem l, we have n_l training samples (x_i^l, y_i^l) indexed by $i \in \{1, \cdots, n_l\}$, $\bar{n}_l$ validation samples (x_j^l, y_j^l) indexed by $j \in \{1, \cdots, \bar{n}_l\}$,
- For each problem, assume that we have a set of candidate hypothesis spaces $\mathcal{H}_{l,\theta}$, where θ is shared among the problems.

Given a fixed structural parameter θ, the predictor for problem l can be estimated using ERM over the hypothesis space $H_{l,\theta}$:

$$\hat{f}_{l,\theta} = \arg \min_{f \in \mathcal{H}_{l,\theta}} \sum_{i=1}^{n_l} L(f(x_i^l), y_i^l)$$

To optimize the structural parameter θ:

$$\theta^* = \arg\min_{\theta} \sum_{l=1}^{m} \frac{1}{\bar{n}_l} \sum_{j=1}^{\bar{n}_l} L(\hat{f}_{l,\theta}(x_j^l), y_j^l)$$

Related work

- θ is hyperparameter:Bilevel Programming for Hyperparameter Optimization and Meta-Learning
- θ is initialization: optimization-based meta learning
- θ is gradient direction: l2l by GD by GD

Back to 2005, such an approach can lead to a quite difficult computational procedure.

A direct solution is to jointly optimize on the training set w.r.t both the predictors $f_{l,\theta}$, and the structural parameter θ:

$$\theta^*, \{f_l\} = \arg \min_{\theta, \{f_l \in H_{l,\theta}\}} \sum_{l=1}^{m} \frac{1}{n_l} \sum_{i=1}^{n_l} L(f_{l,\theta}(x_i^l), y_i^l)$$

Model designing:

$$f_l(x) = {w_l}^T \Phi(x) + {v_l}^T \Psi_\theta(x),$$

θ is the common structure parameter shared by all problems. To simplify the problem, suppose the shared space is linear whose basis is θ:

$$f_\theta(w_l, v_l; x) = {w_l}^T \Phi(x) + {v_l}^T \theta \Psi(x), \quad s.t.\ \theta\theta^T = I$$

Considering a special case where $\Phi(x) = x, \Psi(x) = x$:

$$\theta^*, \{w_l, v_l\} = \arg \min_{\theta, \{w_l, v_l\}} \sum_{l=1}^{m} (\frac{1}{n_l} \sum_{i=1}^{n_l} L((w_l + \theta^T v_l)^T x_i^l, y_i^l) + \lambda_l ||w_l||_2^2),$$

$$s.t.\ \theta\theta^T = I$$

The objective can be minimized by alternative optimization, suppose $u_l = w_l + \theta^T v_l$:

$$\theta^*, \{w_l, v_l\} = \arg \min_{\theta, \{w_l, v_l\}} \sum_{l=1}^{m} (\frac{1}{n_l} \sum_{i=1}^{n_l} L(u_l^T x_i^l, y_i^l) + \lambda_l ||u_l - \theta^T v_l||_2^2)$$

$$s.t.\ \theta\theta^T = I$$

- Fix θ, v_l, optimize u_l by gradient descent.

Fix u_l, and optimize (6) with respect to (θ, v_l).

$$\theta, \{v_l\} = \arg\min_{\theta,\{v_l\}} \sum_l \lambda_l ||u_l - \theta^T v_l||_2^2, \quad s.t.\ \theta\theta^T = I$$

It is equivalent to:

$$\theta = \arg\max \sum_{i=1}^{m} \lambda_l ||\theta u_l||_2^2, \quad s.t.\ \theta\theta^T = I$$

Let $U = [\sqrt{\lambda_1}u_1, \cdots, \sqrt{\lambda_m}u_m]$, we have:

$$\theta = \arg\max_{\theta} tr(\theta U U^T \theta^T), \quad s.t.\ \theta\theta^T = I$$

θ are given by the eigenvalues, $v_l = \theta u_l$.

Experiments

Semi-supervised learning:

- create multiple prediction problems from unlabeled data, and learn θ.
- Learn a predictor for the target problem on the originally labeled data, using θ computed in the first step.

The framework requires auxiliary problems with the following two characteristics:

- Automatic labeling: we need to automatically generate various “labeled” data for the auxiliary problems from unlabeled data.
- Relevancy: auxiliary problems should be related to the target problem (that is, they share a certain predictive structure) to some degree.

We experiment with the following types of auxiliary problems:

- Freq: predicts the most frequent word by observing one half of the words
- Top-k: predicts combinations of the top-k choices of the classifier trained with labeled data
- Multi-k: for the multi-category target task, predicts the top-k choices of the classifier (trained with labeled data), regarding them as multi-category auxiliary labels