Test-Time Training for Generalization under Distribution Shifts

0. Abstract

- improving the performance of predictive models
- when training and test data come from different distributions
- turns a single unlabeled test instance into self-supervised learning problem
- we update the model parameters before making a prediction on this instance

1. Introduction

- supervised learning remains weak at generalization under distribution shifts
- a new take on generalization
- without any mathematical structure or data available at training time to anticipate the distribution shifts
- the unlabeled test instance presented at test-time gives us a hint about the distribution from which it was drawn
- allow model parameters to depend on the test instance
- variable decision boundary $\theta(x)$
- create a self-supervised learning problem based on this single test instance, updating model parameters at test-time before making a prediction
- the task rotates an image and assigns the angle as the label

2. Method

Consider a standard K-layer neural network

parameters θ_k for layer k

the stacked parameter vector $\theta = (\theta_1, \cdots, \theta_K)$

loss function $l_m(x, y; \theta)$ on the test instance (x, y)

training data $(x_1, y_1), \cdots, (x_n, y_n)$ drawn i.i.d from a distribution P

Standard empirical risk minimization corresponds to solving optimization problem:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} l_m(x_i, y_i; \theta)$$

requires a self-supervised auxiliary task with loss function $l_s(x)$: rotation prediction task (effective feature learning, four-way classification problem)

the auxiliary task shares some of the model parameters $\theta_e = (\theta_1, \cdots, \theta_k)$ up to a certain $k \in \{1, \cdots, K\}$: shared feature extractor

auxiliary task specific parameters $\theta_s = (\theta'_{k+1}, \cdots, \theta'_K)$: self-supervised task branch

$\theta_m(\theta_{k+1}, \cdots, \theta_K)$: the main task branch

training is done in the fashion of multi-task learning

- the model is trained on both tasks on the same data drawn from P
- losses for both tasks are added together
- gradients are taken for the collection of all parameters

"the joint training problem"

$$\min_{\theta_e, \theta_m, \theta_s} \frac{1}{n} \sum_{i=1}^{n} l_m(x_i, y_i; \theta_m, \theta_e) + l_s(x_i; \theta_s, \theta_e)$$

test-time training fine-tunes the shared feature extractor

- by minimizing the auxiliary task loss on x

$$\min_{\theta_e} l_s(x; \theta_s, \theta_e)$$

θ_e^* the (approximate) minimizer

model then makes a prediction using the updated parameters $\theta(x) = (\theta_e^*, \theta_m)$

after making prediction on x, θ_e^* is discarded.

<when the test instances arrive online sequentially>

θ is intialized with $\theta(x_{t-1})$

3. Empirical Results

Network details : for CIFAR-10 (ResNets 26-layer), for ImageNet (ResNets 18-layer)

Optimization details: we use stochastic gradient descent (SGD) with weight decay and momentum; learning rate starts at 0.1 and is dropped by a factor of ten at two scheduled milestones, to 0.01 and 0.001