Machine Learning Fall 2019 Final Project
First Week Report

Zhuchkov Alexey BS17 -DS-01

Introduction

This report analyzes and summarizes the implementation of Machine learning assignment 3. This assignment concentrated on solving Domain Adaptation task and the report gives a description of a baseline model for it.

In the real world, it's a quite common situation when the domain of the input data changed while the task domain (the labels) remained the same. The input and task distributions could differ at the same time. In these cases, domain adaptation comes to rescue. Domain adaptation is a sub-discipline of machine learning which deals with scenarios in which a model trained on a source distribution is used in the context of a different (but related) target distribution. In general, domain adaptation uses labeled data in one or more source domains to solve new tasks in a target domain. The level of relatedness between the source and target domains hereby usually determines how successful the adaptation will be.

The datasets for training and testing were SVHN and MNIST accordingly. The model explanation, train and test accuracies and experiments with different techniques will be given in the following sections.

Models results

I've decided to try different techniques to try to solve the problem of domain adaptation. There are several models with different setups:
Optimizer: SGD (learning rate=0.05, momentum = 0.3) or Adam (torch default initial parameters)
Dropout: presence/absence
Batch normalisation: presence/absence

The final model uses Adam; dropout and batch normalization are absent. It gives the best result of accuracy (62.39%) on unseen MNIST data. Number of epochs - 3

Model	Optimizer	Batch norm	Dropout	SVHN-train	SVHN-test	MNIST-test
0	SGD	+	+	85.14%	83.44%	56.43%
1	SGD	-	+	85.63%	83.82%	54.67%
2	SGD	+	-	90.63%	87.45%	59.18%
3	SGD	-	-	89.51%	86.21%	57.80%
4	Adam	+	+	86.59%	84.94%	56.11%
5	Adam	-	+	85.63%	83.82%	59.54%
6	Adam	+	-	90.95%	87.45%	48.63%
Final Model	Adam	-	-	89.43%	86.44%	62.39%

*green - the best, red - the worst

Final Model architecture

№	Layer	Input	Output	Kernel
1	Conv2D	1 x 28 x 28	10 x 24 x 24	5 x 5
2	MaxPool2D	10 x 24 x 24	10 x 12 x 12	2 x 2
3	ReLU	10 x 12 x 12	10 x 12 x 12	-
4	Conv2D (without batch normalization and dropout)	10 x 12 x 12	20 x 8 x 8	5 x 5
5	MaxPool2D	20 x 8 x 8	20 x 4 x 4	2 x 2
6	ReLU	20 x 4 x 4	20 x 4 x 4	-
7	FC_1 (with ReLU activation)	320	100	-
8	FC_2 (with softmax)	100	10	-
Total: 8 layers				

Model explanation and results discussion

As preprocessing techniques resize of the images from SVHN to an MNIST size (28 x 28), normalization of the images and transforming both datasets to grayscale were used. The template of the code has been taken from Lab 11 and processed. Code from lab worked well on MNIST and SVHN separately, the goal has been to improve results of the model in case of domain shift. Tests with different techniques showed the next results.

Analyzing the output results corresponding to each epoch I realised that model with Adam converges faster in terms of epochs. Batch normalization and dropout should prevent the model from overfitting. However, the model without addons has better results. Also, batch normalization should help the model to adapt well to new unseen distribution - MNIST (solves a bit domain adaptation problem). Nevertheless, it does not help well to model with Adam optimizer, only in the SGD case. Training models with Adam optimizer have good results on SVHN sets and well on MNIST-test. However, it cannot adapt easily to another dataset. To sum up, the result of all 8 models is close to each other, but the model with Adam optimizer and no dropout or batch normalization has the best result.

The results with regard to domain adaptation

The described actions were taken in order to make the model more domain-invariant: when new data previously not seen by the model is given as input, the accuracy of a model can decline significantly. It happens because the model has overfitted for a specific dataset but failed to generalize to another one. The experiments I have conducted with different techniques such as dropout and batch normalization, as well as different optimizers (Adam and SGD) have led to the model able to predict the class of target domain with accuracy more than 60%.

References

1. Base template (Lab 11 CNN)
2. The optimal number of epochs
3. Dropout and normalization ordering
4. Dropout in NN
5. PyTorch documentation
6. Domain adaptation in CV
7. CNN and layers