Semi-Supervised Learning with Deep Generative Models

S. Saemundsson, Y. Gao, S. Popescu

Department of Engineering, University of Cambridge

Objectives

The motivation behind reproducing the chosen paper was a better understanding of semi-supvervised learning methods and stochastic variational inference. At the outset of the project our goals were as follows:

- Implement our own version of the models found in (1) using TensorFlow (2).
- Evaluate their performance on the MNIST dataset and contrast with the original paper.
- Ultimately the goal was to reproduce the results obtained using the stacked deep-generative model of (1) with only 100 labelled examples.

Introduction

Ever-increasing unlabelled data together with prohibitive costs of manually labelling it calls for models that are able to efficiently and accurately generalise from a small number of examples. The framwork described in (1) formulates the semi-supervised problem as a generative one and uses deep neural networks to parameterise the densities of the data. It also provides a scalable method for stochastic variational inference for joint optimisation of model and variational parameters. Quantitative results show that this approach improves on classification accuracy in benchmark problems in the semi-supervised domain and qualitatively the models are shown to separate the data based on both data classes as well as intra-class variability. Possible domains with high-impact from semi-supervised learning techniques range from speech analysis and natural language parsing to genomics, research areas predominantely abundant with unlabeled data but with high labeling costs.

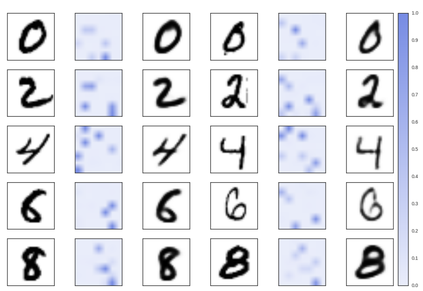

Figure 1: Examples of `flattened` 49 dimensional latent representations of the 784 dimensional images in the MNIST dataset. The columns are as follows: Original → Latent → Reconstructed with two examples each.

Latent Discriminative Model (M1)

Facing a scarcity of labels, the most common starting point is to construct an embedding layer with the main goal of clustering related observations ina latent feature space that permits high accuracy classifciation. The novelty of the current method resides in using a deep generative model for feature representation as opposed to more conventional linear embedding or features derived from a standard autoencoder. The generative model has the following form:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I}); \qquad p_\theta(\boldsymbol{x}|\boldsymbol{z}) = f(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}) \tag{1}$$

where $f(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta})$ is a neural network, $\boldsymbol{x}$ the observations and $\boldsymbol{z}$ the latent representations.

Figure 2: Analogies learnt by the M2 model. The leftmost label and image are used to infer a latent representation under the trained model. Keeping the latent representation fixed and varying the label, the generative model produces images with similar styles and orientation.

Generative Semi-Supervised Model (M2)

M2 describes the data as being generated by a latent class variable y and another one $\boldsymbol{z}$. The model has the following form:

$$p(y) = \text{Cat}(y|\pi);\ p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I}) \tag{8}$$

$$p_\theta(\boldsymbol{x}|y, \boldsymbol{z}) = f(\boldsymbol{x}|y, \boldsymbol{z}, \boldsymbol{\theta}) \tag{9}$$

Where $\text{Cat}(y|\pi)$ is the categorical distribution and $f(\boldsymbol{x}|y, \boldsymbol{z}, \boldsymbol{\theta})$ a neural network. Classification is thus performed by approximate inference of the latent variable y.

Stacked Model (M1+M2)

A deep generative model with two layers of stochastic variables can be established by combining model 1 and model 2:

$$p(\boldsymbol{x}, y, \boldsymbol{z}_1, \boldsymbol{z}_2) = p(y)p(\boldsymbol{z}_2)p_\theta(\boldsymbol{z}_1|y, \boldsymbol{z}_2)p_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \tag{2}$$

where the priors are the same as in the individual models and both $p_\theta(\boldsymbol{z}_1|y, \boldsymbol{z}_2)$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z}_1)$ are parameterised by neural networks. In practise this is done by first learning a latent representation using `M1` which is used as input into the neural network for `M2`.

Variational Inference

In all our models, computation of the exact posterior distribution is intractable due to the nonlinear, non-conjugate dependencies between the random variables. Therefore using variational inference is an essential approach to allow for tractable and scalable inference and parameter learning. This is done using a variational approximation of the posterior $q_\phi(\cdot)$. The objective to minimise in the case of `M1` can be shown to be:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_q\left(\log p_\theta(\mathbf{x}|\boldsymbol{z})\right]) - \text{KL}\left(q_\phi(\boldsymbol{z}|\mathbf{x})||p_\theta(\boldsymbol{z})\right) \tag{3}$$

In the case of `M2` there are two distinct types of datapoints (i.e. labelled and unlabelled):

$$\begin{aligned}&\log p_\theta(\mathbf{x}, y) \geq -\mathcal{L}(\mathbf{x}, y) = \\ &\mathbb{E}_q\left(\log p_\theta(\mathbf{x}|\boldsymbol{z}, y) + \log p_\theta(y) + \log p(\boldsymbol{z}) - q_\phi(\boldsymbol{z}|\mathbf{x}, y)\right)\end{aligned} \tag{4}$$

$$\log p_\theta(\mathbf{x}) \geq -\mathcal{U}(\boldsymbol{x}) = \sum_y q_\phi(y|\boldsymbol{x}))(-\mathcal{L}(\mathbf{x}, y)) + \mathcal{H}(q_\phi(y|\boldsymbol{x})) \tag{5}$$

The bound on the marginal likelihood for the entire data is then:

$$\mathcal{J} = \sum_{\boldsymbol{x}, y} \mathcal{L}(\mathbf{x}, y) + \sum_{\boldsymbol{x}} \mathcal{U}(\boldsymbol{x}) \tag{6}$$

Additionally, in order for the predictive distribution $q_\phi(y|\boldsymbol{x})$ to contribute also to the term pertaining to the unlabelled data, a hyperparameter α was introduced whereby:

$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \cdot \mathbb{E}_{p(\boldsymbol{x}, \boldsymbol{y})}(-\log q_\phi(y|\boldsymbol{x})) \tag{7}$$

This hyperparameter can be shown to be equivalent to placing a symmetric Dirichlet prior over the $\boldsymbol{\pi}$ parameters of the categorical distribution.

Results

Table 1 shows the accuracy obtained using **M1+M2** on the **MNIST** dataset using 100, 1000, 3000 labelled examples in training.

#Labelled	Accuracy	Original
100	95.2 (±0.4)	96.67 (±0.14)
1000	96.7 (±0.1)	97.41 (±0.05)
3000	97.1 (±0.1)	97.60 (±0.02)

Table 1: Accuracy of M1+M2 on MNIST compared to the original paper.

Experiments

The dataset was split into 55000/5000/10000 datapoints for training, validation and evaluation respectively. Each result with a set number of labelled examples was run 3 times for 1200 epochs of the data using different initialisations to obtain the average in Table 1. The data was further split into 100 batches containing a fixed number of labelled examples (e.g. 10 in each batch for a 1000 labelled examples). The networks in `M1` had two fully connected 500 unit hidden layers. In `M2` there was only one hidden layer for each network with 500 hidden units. In both cases the hidden units had softplus activations.
Gradient ascent was performed using ADAM (3) with a learning rate of 3×10^{-4}, and decay rate parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. The weights were initialised at random and in the case of `M1`, L2 regularisation was applied with weight parameter 0.001. For `M2` the weights were conditioned with a standard normal gaussian prior.

Discussion

The results in Table 1 are slightly lower than that of the original paper. The discrepancy is likely caused by one of two factors: i) difference in implementation (e.g. L2 regularisation), ii) due to time constraints the models were run for only 1200 epochs until *approximate* convergence. TensorFlow was found to be highly useful in terms of balancing *out-of-the-box* components and flexibility which allowed for quick prototyping.

References

(1). Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling. **Semi-supervised Learning with Deep Generative Models**. arXiv:1406.5298, 2014. (2). Google Research. **TensorFlow**: Large-Scale Machine Learning on Heterogeneous Distributed Systems. `http://www.tensorflow.org`. (3). Diederik P. Kingma, Jimmy Lei Ba. **ADAM: A Method for Stochastic Optimization**. arxiv:1412.6980, 2014