Online Incremental Feature Learning with Denoising Autoencoders—summary

Matěj Nikl

August 6, 2016

Determining the optimal model complexity, i.e. the number of features for DAE is a nontrivial problem. Moreover, in large-scale and online datasets the cross-validation method enabling the hyper-parameter search is even more challenging.

This adaptive feature learning algorithm tries to address the given issue by:

1. adding and training new features to minimize the residual of the objective function
2. merging similar features to avoid redundant feature representations

Having a variable number of features gives the model the ability to make itself more/less complex according to the (even changing) dataset needs.

1 Denoising Autoencoder

The standard Denoising Autoencoder is used as a building block for incremental feature learning, it consists of four components:

- a conditional distribution $q(\widetilde{\mathbf{x}}|\mathbf{x})$, that corrupts $\mathbf{x}$ into $\widetilde{\mathbf{x}}$
- an encoding function $f(\widetilde{\mathbf{x}}) = \sigma(\mathbf{W}\widetilde{\mathbf{x}} + \mathbf{b}) \equiv \mathbf{h} \in \mathbb{R}^N$, which gives a representation of input data $\mathbf{x}$
- an decoding function $g(\mathbf{h}) = \sigma(\mathbf{W}^T\mathbf{h} + \mathbf{c}) \equiv \widehat{\mathbf{x}} \in \mathbb{R}^D$, which recovers the data from the representation $\mathbf{h}$
- a differentiable (cross-entropy) cost function $\mathcal{L}(\mathbf{x}) = \mathcal{H}(\mathbf{x}, \widehat{\mathbf{x}}) \in \mathbb{R}$, which computes the dissimilarity between input and reconstruction

2 Incremental Feature Learning

The incremental feature learning algorithm needs to address the following problems:

1. **when** to add and merge features
2. **how** to add and merge features

In order to deal with these problems, the authors have chosen the following procedure:

1. form a set B of hard training examples: $B \leftarrow B \cup \{\mathbf{x}\}$ if $\mathcal{L}(\mathbf{x}) > \mu$, which is used to greedily train the new features
2. merge similar + add new features when there are enough examples in B: $|B| > \tau$

I.e. when there is enough *hard* examples, $2\Delta M$ old features are merged into ΔM new features, ΔN new features are simply added, and these new features are greedily trained on the set B of hard training examples. After convergence, set $B = \emptyset$ and continue standard training of all features.

In this paper μ was set as average of the objective function for recent 10000 examples, τ was also set as 10000.

Notations. D denotes input dimension, N the number of (existing) features, and K the number of class labels. The parameters for generative training are defined as the (tied) weight matrix $\mathbf{W} \in \mathbb{R}^{N \times D}$, the hidden bias $\mathbf{b} \in \mathbb{R}^N$, and the input bias $\mathbf{c} \in \mathbb{R}^D$. Similarly, the parameters for discriminative training are composed of the weight matrix $\mathbf{\Gamma} \in \mathbb{R}^{K \times N}$ between hidden and output units, the output bias $\boldsymbol{\nu} \in \mathbb{R}^K$, as well as the weight matrix $\mathbf{W}$ and the hidden bias $\mathbf{b}$. $\mathcal{N}$ is used to denote *new* features and $\mathcal{O}$ to denote existing or *old* features, thus $\theta_\mathcal{O} = \{\mathbf{W}_\mathcal{O}, \mathbf{b}_\mathcal{O}, \mathbf{c}, \mathbf{\Gamma}_\mathcal{O}, \boldsymbol{\nu}\}$, and $\theta_\mathcal{N} = \{\mathbf{W}_\mathcal{N}, \mathbf{b}_\mathcal{N}, \mathbf{c}, \mathbf{\Gamma}_\mathcal{N}, \boldsymbol{\nu}\}$.

2.1 Adding Features

New features are being added and trained upon, whenever there is enough examples in the *hard set* B. The basic idea for efficient training is that only the new features $\theta_\mathcal{N}$ are trained to minimize the objective function while keeping $\theta_\mathcal{O}$ fixed.

2.1.1 Generative Objective Function

Measures reconstruction error between the input $\mathbf{x}$ and the reconstruction $\widehat{\mathbf{x}}$:

$$\mathcal{L}_{\texttt{gen}}(\mathbf{x}) = \mathcal{H}(\mathbf{x}, \widehat{\mathbf{x}}) \tag{1}$$

The encoding and decoding functions for the new features can be written as:

$$\mathbf{h}_\mathcal{N} = sigm(\mathbf{W}_\mathcal{N}\widetilde{\mathbf{x}} + \mathbf{b}_\mathcal{N}) \tag{2}$$

$$\widehat{\mathbf{x}} = sigm(\mathbf{W}_\mathcal{N}^T\mathbf{h}_\mathcal{N} + \mathbf{W}_\mathcal{O}^T\mathbf{h}_\mathcal{O} + \mathbf{c}) \tag{3}$$

Moreover, since $\theta_\mathcal{O}$ is fixed, equation (3) can be rewritten to:

$$\widehat{\mathbf{x}} = sigm(\mathbf{W}_\mathcal{N}^T\mathbf{h}_\mathcal{N} + c_d(\mathbf{h}_\mathcal{O})) \tag{4}$$

where $c_d(\mathbf{h}_\mathcal{O}) \equiv \mathbf{W}_\mathcal{O}^T\mathbf{h}_\mathcal{O} + \mathbf{c}$ is viewed as a *dynamic* decoding bias. This allows for a efficient way of training

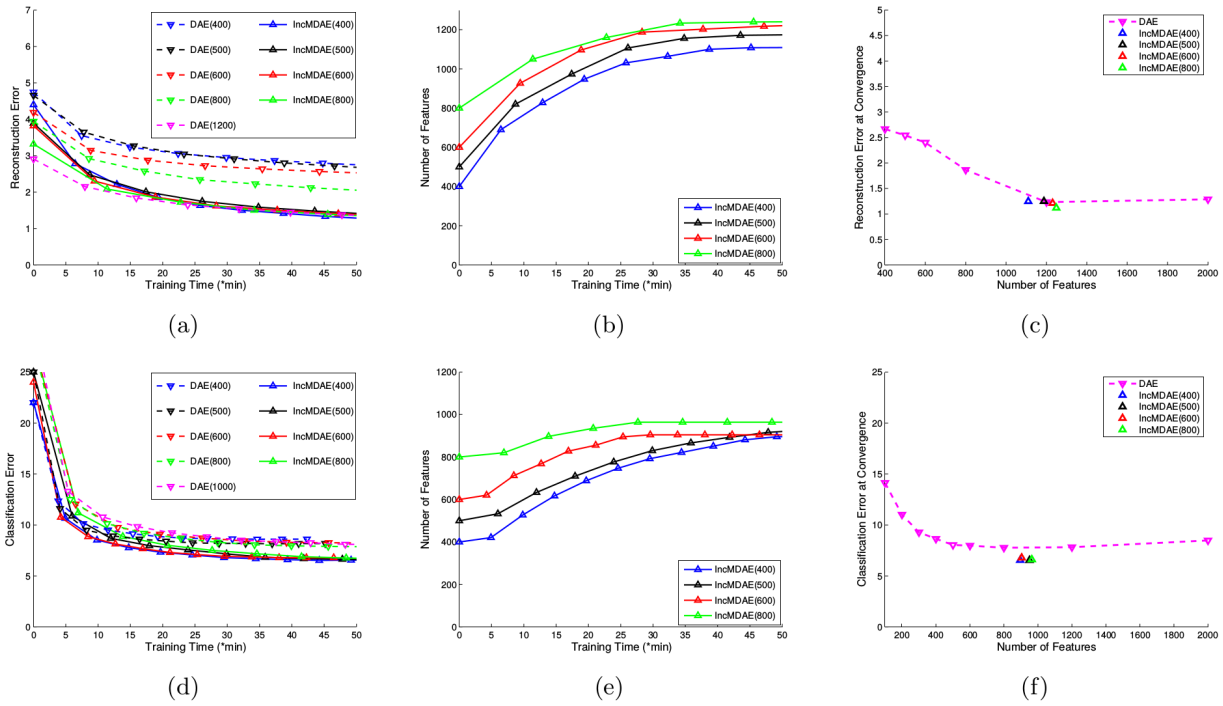

Figure 1: (a,d) Test (online) reconstruction error and classification error of incremental (IncMDAE) and non-incremental DAE. (b,e) The number of features for incremental learning with generative and hybrid criteria. (c,f) Test reconstruction error and classification error at convergence. The numbers in parentheses refer to the initial number of features.

new features, because $c_d(\mathbf{h}_\mathcal{O})$ can be computed for each training example only once and then recalled without additional computational cost.

2.1.2 Discriminative Objective Function

Computes a classification loss between the actual label $\mathbf{y}$ and the predicted label $\widehat{\mathbf{y}}$:

$$\mathcal{L}_{\texttt{disc}}(\mathbf{x}, \mathbf{y}) = \mathcal{H}(\mathbf{y}, \widehat{\mathbf{y}}(\mathbf{x})) \tag{5}$$

$$\widehat{\mathbf{y}} = \text{softmax}(\mathbf{\Gamma}_\mathcal{N} f_\mathcal{N}(\widetilde{\mathbf{x}}) + \mathbf{\Gamma}_\mathcal{O} f_\mathcal{O}(\widetilde{\mathbf{x}}) + \boldsymbol{\nu}) \tag{6}$$

A similar interpretation for $\mathbf{\Gamma}_\mathcal{O} f_\mathcal{O}(\widetilde{\mathbf{x}}) + \boldsymbol{\nu}$, as in the generative training is possible, and therefore, the new parameters $\{\mathbf{W}_\mathcal{N}, \mathbf{b}_\mathcal{N}, \mathbf{\Gamma}_\mathcal{N}\}$ can be trained efficiently as well.

2.1.3 Hybrid Objective Function

The proposed objective function for training new features is a linear combination of a generative and a discriminative objective function, called a *hybrid* objective function:

$$\mathcal{L}_{\texttt{hybrid}} = \mathcal{L}_{\texttt{disc}}(\mathbf{x}, \mathbf{y}) + \lambda \mathcal{L}_{\texttt{gen}}(\mathbf{x}) \tag{7}$$

The values $\lambda \in [0.1, 0.4]$ gave roughly the best performance.

2.2 Merging Features

As a counterpart to the feature adding is the process of *merging* the features, because only adding would lead to redundant features and overfitting. Feature merging is done in two steps:

- select a pair of candidate features to be merged, $\mathcal{M} = \{m_1, m_2\} \subset \mathcal{O}$, whose cosine distance $d(\mathbf{W}_{m_1}, \mathbf{W}_{m_2})$ is minimal, and replace $f_\mathcal{O}$ by $f_\mathcal{O} \setminus \mathcal{M}$
- add a new feature mapping $f_\mathcal{N}$ that *replaces* the merged features, while initializing it as a weighted average of the selected two for faster convergence:

$$\theta_\mathcal{N} = \frac{\sum_{\mathbf{x} \in B} P(h_{m_1} | \mathbf{x}; \theta_{m_1}) \theta_{m_1} + P(h_{m_1} | \mathbf{x}; \theta_{m_2}) \theta_{m_2}}{\sum_{\mathbf{x} \in B} P(h_{m_1} | \mathbf{x}; \theta_{m_1}) + P(h_{m_1} | \mathbf{x}; \theta_{m_2})} \tag{8}$$

2.3 Selecting ΔN and ΔM

Some kind of metric must be used to control the number of features being merged ($2\Delta M$) and added (ΔN). A combination of AIC and BIC was used by authors, however the performance was fairly robust to the choice of update rule for ΔN and ΔM.

3 Principles of modularization

Individual hidden units are treated as modules – they are being manipulated (added and merged), however this does not allow for any kind of modular-like design for multi-task setting.

4 Principles of growing

This kind of architecture is well capable of growing (and shrinking), as explained in 2.1 and 2.2, however it only increases or decreases the model's complexity (and allows to adapt faster to e.g. changing distribution of the dataset during online training).