On autoencoder scoring

Hanna Kamyshanska KAMYSHANSKA@FIAS.UNI-FRANKFURT.DE
Goethe Universität Frankfurt, Robert-Mayer-Str. 11-15, 60325 Frankfurt, Germany

Roland Memisevic MEMISEVR@IRO.UMONTREAL.CA
University of Montreal, CP 6128, succ Centre-Ville, Montreal H3C 3J7, Canada

Abstract

Autoencoders are popular feature learning models because they are conceptually simple, easy to train and allow for efficient inference and training. Recent work has shown how certain autoencoders can assign an unnormalized "score" to data which measures how well the autoencoder can represent the data. Scores are commonly computed by using training criteria that relate the autoencoder to a probabilistic model, such as the Restricted Boltzmann Machine. In this paper we show how an autoencoder can assign meaningful scores to data independently of training procedure and without reference to any probabilistic model, by interpreting it as a dynamical system. We discuss how, and under which conditions, running the dynamical system can be viewed as performing gradient descent in an energy function, which in turn allows us to derive a score via integration. We also show how one can combine multiple, unnormalized scores into a generative classifier.

1. Introduction

Unsupervised learning has been based traditionally on probabilistic modeling and maximum likelihood estimation. In recent years, a variety of models have been proposed which define learning as the construction of an unnormalized energy surface and inference as finding local minima of that surface. Training such *energy-based models* amounts to decreasing energy near the observed training data points and increasing it everywhere else (Hinton, 2002; lec). Maximum likelihood learning can be viewed as a special case, where the exponential of the negative energy integrates to 1.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

The probably most successful recent examples of non-probabilistic unsupervised learning are autoencoder networks, which were shown to yield state-of-the-art performance in a wide variety of tasks, ranging from object recognition and learning invariant representations to syntatic modeling of text (Le et al., 2012; Socher et al., 2011; Rolfe & LeCun, 2013; Swersky et al., 2011; Vincent, 2011; Memisevic, 2011; Zou et al., 2012). Learning amounts to minimizing reconstruction error using back-prop. Typically, one regularizes the autoencoder, for example, by adding noise to the input data or by adding a penalty term to the training objective (Vincent et al., 2008; Rifai et al., 2011).

The most common operation after training is to infer the latent representation from data, which can be used, for example, for classification. For the autoencoder, extracting the latent representation amounts to evaluating a feed-forward mapping. Since training is entirely unsupervised, one autoencoder is typically trained for all classes, and it is the job of the classifier to find and utilize class-specific differences in the representation. This is in contrast to the way in which probabilistic models have been used predominantly in the past: Although probabilistic models (such as Gaussian mixtures) may be used to extract class-independent features for classification, it has been more common to train one model per class and to subsequently use Bayes' rule for classification (Duda & Hart, 2001). In the case where an energy-based model can assign *confidence scores* to data, such class-specific unsupervised learning is possible without Bayes' rule: When scores do not integrate to 1 one can, for example, train a classifier on the vector of scores across classes (Hinton, 2002), or back-propagate label information to the class-specific models using a "gated softmax" response (Memisevic et al., 2011).

1.1. Autoencoder scoring

It is not immediatly obvious how one may compute scores from an autoencoder, because the energy landscape does not come in an explicit form. This is in contrast to undirected probability models like the Restricted Boltzmann Machine (RBM) or Markov Random Fields, which *define* the score (or negative energy) as an unnormalized probability distribution. Recent work has shown that autoencoders can assign scores, too, if they are trained in a certain way: If noise is added to the input data during training, minimizing squared error is related to performing *score matching* (Hyvärinen, 2005) in an undirect probabilistic model, as shown by (Swersky et al., 2011) and (Vincent, 2011). This, in turn, makes it possible to use the RBM energy as a score. A similar argument can be made for other, related training criteria. For example, (Rifai et al., 2011) suggest training autoencoders using a "contraction penalty" that encourages latent representations to be locally flat, and (Alain & Bengio, 2013) show that such regularization penalty allows us to interpret the autoencoder reconstruction function as an estimate of the gradient of the data log probability.[1]

All these approaches to defining confidence scores rely on a regularized training criterion (such as denoising or contraction), and scores are computed by using the relationship with a probabilistic model. As a result scores can be computed easily only for autoencoders that have sigmoid hidden unit activations and linear outputs, and that are trained by minimizing squared error (Alain & Bengio, 2013). The restriction of activation function is at odds with the growing interest in unconventional activation functions, like quadratic or rectified linear units which seem to work better in supervised recognition tasks (eg., (Krizhevsky et al., 2012)).

In this work, we show how autoencoder confidence scores may be derived by interpreting the autoencoder as a *dynamical system.* The view of the autoencoder as a dynamical system was proposed by (Seung, 1998), who also demonstrated how de-noising as a learning criterion follows naturally from this perspective. To compute scores, we will assume "tied weights", that is, decoder and encoder weights are transposes of each other. In contrast to probabilistic arguments based on score matching and regularization (Swersky et al., 2011; Vincent, 2011; Alain & Bengio, 2013), the dynamical systems perspective allows us to assign confidence scores to networks with sigmoid output units (binary data) and arbitrary hidden unit activations (as long as these are integrable). In contrast to (Rolfe & LeCun, 2013), we do not address the role of dynamics in learning. In fact, we show how one may derive confidence scores that are *entirely agnostic* to the learning procedure used to train the model.

[1]The term "score" is also frequently used to refer to the gradient of the data log probability. In this paper we use the term to denote a confidence value that the autoencoder assigns to data.

As an application of autoencoder confidence scores we describe a generative classifier based on class-specific autoencoders. The model achieves 1.27% error rate on permutation invariant MNIST, and yields competitive performance on the deep learning benchmark dataset by (Larochelle et al., 2007).

2. Autoencoder confidence scores

Autoencoders are feed forward neural networks used to learn representations of data. They map input data to a hidden representation using an encoder function

$$h\big(W\boldsymbol{x} + \boldsymbol{b}_h\big) \tag{1}$$

from which the data is reconstructed using a linear decoder

$$r(\boldsymbol{x}) = Ah\big(W\boldsymbol{x} + \boldsymbol{b}_h\big) + \boldsymbol{b}_r \tag{2}$$

We shall assume that $A = W^{\mathrm{T}}$ in the following ("tied weights"). This is common in practice, because it reduces the number of parameters and because related probabilistic models, like the RBM, are based on tied weights, too.

For training, one typically minimizes squared reconstruction error $(r(\boldsymbol{x}) - \boldsymbol{x})^2$ for a set of training cases. When the number of hidden units is small, autoencoders learn to perform dimensionality reduction. In practice, it is more common to learn sparse representations by using a large number of hidden units and training with a regularizer (eg., (Rifai et al., 2011; Vincent et al., 2008)). A wide variety of models can be learned that way, depending on the activation function, number of hidden units and nature of the regularization during training. The function $h(\cdot)$ can be the identity or it can be an element-wise non-linearity, such as a sigmoid function. Autoencoders defined using Eq. 2 with tied weights and logistic sigmoid nonlinearity $h(a) = \big(1 + \exp(-a)\big)^{-1}$ are closely related to RBMs (Swersky et al., 2011; Vincent, 2011; Alain & Bengio, 2013), one can assign confidence scores to data in the form of unnormalized probabilities.

For binary data, the decoder typically gets replaced by

$$r(\boldsymbol{x}) = \sigma\big(Ah\big(W\boldsymbol{x} + \boldsymbol{b}_h\big) + \boldsymbol{b}_r\big) \tag{3}$$

and training is done by minimizing cross-entropy loss. Even though confidence scores (negative free energies) are well-defined for binary output RBMs, there has been no analogous score function for the autoencoder, because the relationships with score matching breaks down in the binary case (Alain & Bengio, 2013). As we shall show, the perspective of dynamical systems allows us to attribute the missing link to the lack of symmetry. We also show how we can regain symmetry and thereby obtain a confidence score for binary output autoencoders by applying a log-odds transformation on the outputs of the autoencoder.

2.1. Reconstruction as energy minimization

Autoencoders may be viewed as dynamical systems, by noting that the function $r(\boldsymbol{x}) - \boldsymbol{x}$ (using the definition in Eq. 2) is a *vector field* which represents the linear transformation that $\boldsymbol{x}$ undergoes as a result of applying the reconstruction function $r(\boldsymbol{x})$ (Seung, 1998; Alain & Bengio, 2013). Repeatedly applying the reconstruction function (possibly with a small inference rate ϵ) to an initial $\boldsymbol{x}$ will trace out a non-linear trajectory $\boldsymbol{x}(t)$ in the data-space.

If the number of hidden units is smaller than the number of data dimensions, then the set of fixed points of the dynamical system will be approximately a low-dimensional manifold in the data-space (Seung, 1998). For overcomplete hiddens it can be a more complex structure.

(Alain & Bengio, 2013), for example, show that for denoising and contractive autoencoder, the reconstruction is proportional to the derivative of the log probability of $\boldsymbol{x}$:

$$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} \quad = \quad \lambda \frac{\partial \log P(\boldsymbol{x})}{\partial(\boldsymbol{x})} + o(\lambda),\ \lambda \to 0 \qquad (4)$$

Running the autoencoder by following a trajectory as prescribed by the vector field may also be viewed in analogy to running a Gibbs sampler in an RBM, where the fixed points play the role of a maximum probability "ridge" and where the samples are deterministic not stochastic.

Some vector fields can be written as the derivative of a scalar field: In such a case, running the dynamical system can be thought of as performing gradient descent in the scalar field. We may call this scalar field *energy* $E(\boldsymbol{x})$ and interpret the vector field as a corresponding "force" in analogy to physics, where the potential force acting on a point is the gradient of the potential energy at that point. The autoencoder reconstruction may thus be viewed as pushing data samples in the direction of lower energy (Alain & Bengio, 2013).

The reason why evaluating the potential energy for the autoencoder would be useful is that it allows us to asses how much the autoencoder "likes" a given input $\boldsymbol{x}$ (up to a normalizing constant which is the same for any two inputs). That way, the potential energy plays an analogous role to the free energy in an RBM (Hinton, 2002; Swersky et al., 2011). As shown by (Alain & Bengio, 2013), the view of the autoencoder as modeling an energy surface implies that reconstruction error is *not* a good measure of confidence, because the reconstruction error will be low at both local minima *and* local maxima of the energy.

A simple condition for a vector field to be a gradient field is given by Poincare's *integrability criterion*: For some open, simple connected set U, a continuously differentiable function $F : U \to R^n$ defines a gradient field if and only if

$$\frac{\partial F_j(\boldsymbol{x})}{\partial x_i} = \frac{\partial F_i(\boldsymbol{x})}{\partial x_j}, \quad \forall i, j = 1..n \qquad (5)$$

In other words, integrability follows from symmetry of the partial derivatives.

Consider an autoencoder with shared weight matrix W and biases $\boldsymbol{b}_h$ and $\boldsymbol{b}_r$, which has some activation function $h(.)$ (e.g. sigmoid, hyperbolic tangent, linear). We have:

$$\begin{aligned} \frac{\partial(r_m(\boldsymbol{x}) - x_m)}{\partial x_n} &= \sum_j W_{mj} \frac{\partial h(W\boldsymbol{x} + \boldsymbol{b}_h)}{\partial(W\boldsymbol{x} + \boldsymbol{b}_h)} W_{nj} - 1 \\ &= \frac{\partial(r_n(\boldsymbol{x}) - x_n)}{\partial x_m} \qquad (6) \end{aligned}$$

so the integrability criterion is satisfied.

2.2. Computing the energy surface

One way to find the potential energy whose derivative is $r(\boldsymbol{x}) - \boldsymbol{x}$, is to integrate the vector field (compute its antiderivative).[2] This turns out to be fairly straightforward for autoencoders with a single hidden layer, linear output activations, and symmetric weights, in other words

$$r(\boldsymbol{x}) = W^{\mathrm{T}} h\big(W\boldsymbol{x} + \boldsymbol{b}_h\big) + \boldsymbol{b}_r \qquad (7)$$

[2]After computing the energy function, it is easy to check, in hindsight, whether the vector field defined by the autoencoder is really the gradient field of that energy function: Compute the derivative of the energy, and check if it is equal to $r(\boldsymbol{x}) - \boldsymbol{x}$. For example, to check the correctness of Eqs. 11 and 13 differentiate the equations wrt. $\boldsymbol{x}$.

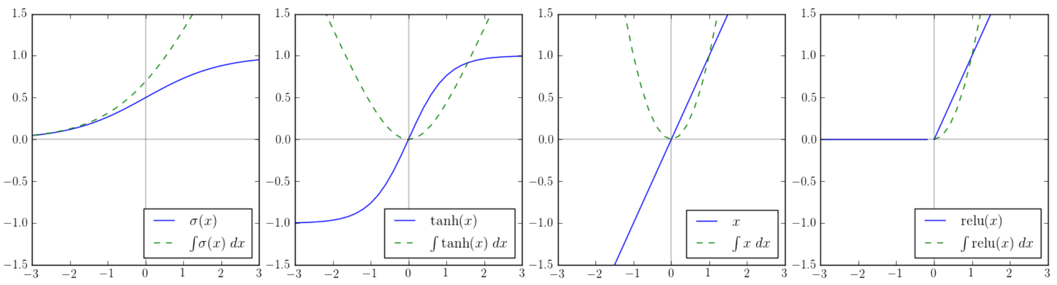

Figure 1. Some hidden unit activation functions and their integrals.

where $h(\cdot)$ is an elementwise activation function, such as the sigmoid. We can now write

$$\begin{aligned} F(\boldsymbol{x}) &= \int (r(\boldsymbol{x}) - \boldsymbol{x})\, d\boldsymbol{x} \\ &= \int \big(W^{\mathrm{T}} h\big(W\boldsymbol{x} + \boldsymbol{b}_h\big) + \boldsymbol{b}_r - \boldsymbol{x}\big)\ d\boldsymbol{x} \qquad (8) \\ &= W^{\mathrm{T}} \int h\big(W\boldsymbol{x} + \boldsymbol{b}_h\big)\, d\boldsymbol{x} + \int (\boldsymbol{b}_r - \boldsymbol{x})\, d\boldsymbol{x} \end{aligned}$$

By defining the auxiliary variables $\boldsymbol{u} = W\boldsymbol{x} + \boldsymbol{b}_h$ and using

$$\frac{d\boldsymbol{u}}{d\boldsymbol{x}} = W^{\mathrm{T}} \Leftrightarrow d\boldsymbol{x} = W^{-\mathrm{T}} d\boldsymbol{u} \qquad (9)$$

we get

$$\begin{aligned} F(\boldsymbol{x}) &= W^{\mathrm{T}} W^{-\mathrm{T}} \int h(\boldsymbol{u})\, d\boldsymbol{u} + \boldsymbol{b}_r{}^{\mathrm{T}} \boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^2 + \text{const} \\ &= \int h(\boldsymbol{u})\, d\boldsymbol{u} + \boldsymbol{b}_r{}^{\mathrm{T}} \boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^2 + \text{const} \\ &= \int h(\boldsymbol{u})\, d\boldsymbol{u} - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \frac{1}{2}\boldsymbol{b}_r{}^{\mathrm{T}}\boldsymbol{b}_r + \text{const} \\ &= \int h(\boldsymbol{u})\, d\boldsymbol{u} - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \text{const} \qquad (10) \end{aligned}$$

where the last equation uses the fact that $\boldsymbol{b}_r$ does not depend on $\boldsymbol{x}$.

If $h(\boldsymbol{u})$ is an elementwise activation function, then the final integral is simply the sum over the antiderivatives of the hidden unit activation functions applied to $\boldsymbol{x}$. In other words, we can compute the integral using the following recipe:

1. compute the net inputs to the hidden units:

$$W\boldsymbol{x} + \boldsymbol{b}_h$$

2. compute hidden unit activations using the antiderivative of $h(\boldsymbol{u})$ as the activation function

3. sum up the activations and subtract $\frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2$

Example: sigmoid hiddens. In the case of sigmoid activation functions $h(\boldsymbol{u}) = (1 + \exp(-\boldsymbol{u}))^{-1}$, we get

$$\begin{aligned} &F_{\text{sigmoid}}(\boldsymbol{x}) \\ &= \int (1 + \exp(-\boldsymbol{u}))^{-1}\, d\boldsymbol{u} - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \text{const} \\ &= \sum_k \log(1 + \exp(W_{\cdot k}^{\mathrm{T}} \boldsymbol{x} + b_{hk})) - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \text{const} \end{aligned} \qquad (11)$$

which is identical to the free energy in a binary-Gaussian RBM (eg., (Welling et al., 2005)). It is interesting to note that in the case of contractive or denoising training (Alain & Bengio, 2013), the energy can in fact be shown to approximate the log-probability of the data (cf., Eq. 4):

$$F(\boldsymbol{x}) := \int (r(\boldsymbol{x}) - \boldsymbol{x})\, d\boldsymbol{x} \propto \log P(\boldsymbol{x}) \qquad (12)$$

But Eq. 11 is more general as it holds independently of the training procedure.

Example: linear hiddens. The antiderivative of the linear activation, $h(\boldsymbol{u}) = \boldsymbol{u}$, is $\boldsymbol{u}^2$, so for PCA and a linear autoencoder, it is simply the norm of the latent representation. More precisely, we have

$$\begin{aligned} F_{\text{linear}}(\boldsymbol{x}) &= \int \boldsymbol{u}\, d\boldsymbol{u} - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \text{const} \\ &= \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u} - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \text{const} \\ &= \frac{1}{2}(W\boldsymbol{x} + \boldsymbol{b}_h)^{\mathrm{T}}(W\boldsymbol{x} + \boldsymbol{b}_h) - \frac{1}{2}\big(\boldsymbol{x} - \boldsymbol{b}_r\big)^2 + \text{const} \end{aligned} \qquad (13)$$

It is interesting to note that, if we disregard biases and assume $WW^{\mathrm{T}} = I$ (the PCA solution), then $F_{linear}(\boldsymbol{x})$ turns into the negative squared reconstruction error.

This is how one would typically assign confidences to PCA models, for example, in a PCA based classfier.

It is straightforward to calculate the energies for other hidden unit activations, including those for which the energy cannot be normalized, in which case there is no corresponding RBM. Two commonly deployed activation functions are, for example: the rectifier, whose antiderivative is the "half-square" $\frac{(sign(\boldsymbol{x})+1)}{2}\boldsymbol{x}^2$, and the softplus activation whose antiderivative is the so-called polylogarithm. A variety of activation functions and their antiderivatives are shown in Fig. 1.

2.3. Binary data

When dealing with binary data it is common to use sigmoid activations on the outputs:

$$r(\boldsymbol{x}) = \sigma\big(W^{\mathrm{T}}h\big(W\boldsymbol{x}+\boldsymbol{b}_h\big)+\boldsymbol{b}_r\big) \tag{14}$$

and training the model using cross-entropy (but like in the case of real outputs, the criterion used for training will not be relevant to compute scores). In the case of sigmoid outputs activations, the integrability criterion (Eq. 6) does not hold, because of the lack of symmetry of the derivatives. However, we can obtain confidence scores by monotonically transforming the vector space as follows: We apply the inverse of the logistic sigmoid (the "log-odds" transformation)

$$\boldsymbol{\xi}(\boldsymbol{x}) = \log\left(\frac{\boldsymbol{x}}{1-\boldsymbol{x}}\right) \tag{15}$$

in the input domain. Now we can define the new vector field

$$\begin{aligned} v(\boldsymbol{x}) &= \boldsymbol{\xi}(r(\boldsymbol{x})) - \boldsymbol{\xi}(\boldsymbol{x}) \\ &= \boldsymbol{\xi}\Big(\sigma\big(W^{\mathrm{T}}h\big(W\boldsymbol{x}+\boldsymbol{b}_h\big)+\boldsymbol{b}_r\big)\Big) - \log\left(\frac{\boldsymbol{x}}{1-\boldsymbol{x}}\right) \\ &= W^{\mathrm{T}}h\big(W\boldsymbol{x}+\boldsymbol{b}_h\big)+\boldsymbol{b}_r - \log\left(\frac{\boldsymbol{x}}{1-\boldsymbol{x}}\right) \end{aligned} \tag{16}$$

The vector field $v(\boldsymbol{x})$ has the same fixed points as $r(\boldsymbol{x})-\boldsymbol{x}$, because invertibility of $\boldsymbol{\xi}(\boldsymbol{x})$ implies

$$r(\boldsymbol{x}) = \boldsymbol{x} \quad\Leftrightarrow\quad \boldsymbol{\xi}(r(\boldsymbol{x})) = \boldsymbol{\xi}(\boldsymbol{x}) \tag{17}$$

So the original and transformed autoencoder converge to the same representations of $\boldsymbol{x}$. By integrating $v(\boldsymbol{x})$, we get

$$\begin{aligned} F(\boldsymbol{x}) = &\int h(\boldsymbol{u})\, d\boldsymbol{u} + {\boldsymbol{b}_r}^{\mathrm{T}}\boldsymbol{x} \\ &- \log\big(1-\boldsymbol{x}\big) - \boldsymbol{x}\log\left(\frac{\boldsymbol{x}}{1-\boldsymbol{x}}\right) + \text{const} \end{aligned} \tag{18}$$

Due to the convention $0\cdot\log 0 = 0$, the second term, $-\log\big(1-\boldsymbol{x}\big) - \boldsymbol{x}\log\left(\frac{\boldsymbol{x}}{1-\boldsymbol{x}}\right)$, vanishes for binary data (Cover & Thomas, 1991). In that case, the energy takes exactly the same form as the free energy of a binary output RBM with binary hidden units (Hinton, 2002). However, hidden unit activation functions can be chosen arbitrarily and enter the score using the recipe described above. Also, training data may not always be binary. When it takes on values between 0 and 1, the log-terms do not equal 0 and have to be included in the energy computation.

3. Combining autoencoders for classification

Being able to assign unnormalized scores to data can be useful in a variety of tasks, including visualization or classification based on ranking of data examples. Like for the RBM, the lack of normalization causes scores to be relative not absolute. This means that we can compare the scores that an autoencoder assigns to multiple data-points but we cannot compare the scores that multiple autoencoders assign to the same data-point. We shall now discuss how we can turn these into a classification decision.

Fig. 2 shows exemplary energies that various types of contractive autoencoder (cAE, (Rifai et al., 2011)), trained on MNISTsmall digits 6 and 9, assign to test cases from those classes. It shows that all models yield fairly well separated confidence scores, when the apropriate anti-derivatives are used. Squared error on the sigmoid networks separates these examples fairly well too (rightmost plot). However, in a multi-class task, using reconstruction error typically does not work well (Susskind et al., 2011), and it is not a good confidence measure as we discussed in Section 2. As we shall show, one can achieve competitive classification performance on MNIST and other data-sets by using properly "calibrated" energies, however.

3.1. Supervised finetuning

Since the energy scores are unnormalized, we cannot use Bayes' rule for classification unlike with directed graphical models. Here, we adopt the approach proposed by (Memisevic et al., 2011) for Restricted Boltzmann Machines and adopt it to autoencoders.

In particular, denoting the energy scores that the autoencoders for different classes assign to data as $E_i(\boldsymbol{x}), i = 1,\ldots,K$, we can define the conditional distribution over classes y_i as the softmax response

$$P(y_i|\boldsymbol{x}) = \frac{\exp(E_i(\boldsymbol{x})+C_i)}{\sum_j \exp(E_j(\boldsymbol{x})+C_j)} \tag{19}$$

where C_i is the bias term for class y_i. Each C_i may be

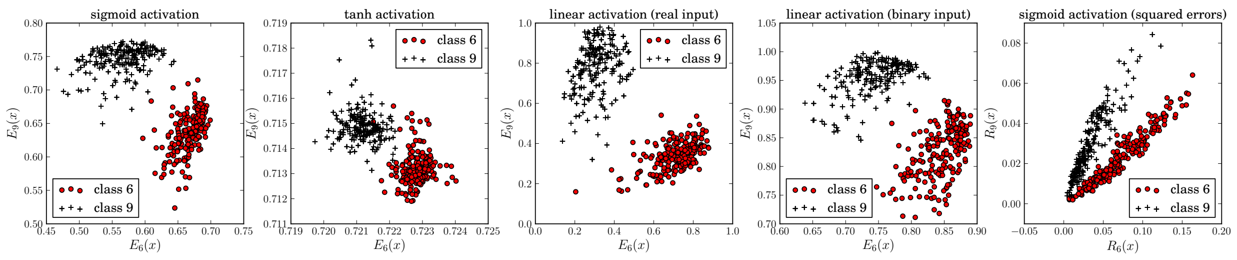

Figure 2. Confidence scores assigned by class-specific contractive autoencoders to digits 6 and 9 from the MNISTsmall data set.

viewed also as the normalizing constant of the i^{th} autoencoder, which, since it cannot be determined from the input data, needs to be trained from the labeled data.

Eq. 19 may be viewed as a "*contrastive*" objective function that compares class y_i against all other classes in order to determine the missing normalizing constants of the individual autoencoders. It is reminiscent of noise-contrastive estimation (Gutmann & Hyvärinen, 2012), with the difference that the contrastive signal is provided by other classes not noise. Optimizing the log-probabilities (log of Eq. 19) is simply a form of logistic regression. We shall refer to the model as autoencoder scoring (AES) in the following. Like the gated softmax model (Memisevic et al., 2011), we may optimize Eq. 19 wrt. the autoencoder parameters by back-propagating the logistic regression cost to the autoencoder parameters.

3.2. Parameter factorization

We showed in Section 2 that scores can be computed for autoencoders with a single hidden layer only. However, if we train multi-layer autoencoders whose bottom layer weights are tied across models, we may view the bottom layers as a way to perform *class-independent pre-processing*. In many classification tasks this kind of pre-processing makes sense, because similar features may appear in several classes, so there is no need to learn them separately for each class. Class-specific autoencoders with shared bottom-layer weights can also be interpreted as standard autoencoders with factorized weight matrices (Memisevic et al., 2011).

Fig. 3 shows an illustration of a factored autoencoder. The model parameters are filter matrices $\boldsymbol{W^x}$ and $\boldsymbol{W^h}$ which are shared among the classes, as well as matrices $\boldsymbol{W^f}, \boldsymbol{B}^h$ and $\boldsymbol{B}^r$ which consist of stacked class-dependent feature-weights and bias vectors. Using one-hot encoded labels, we can write the hidden unit activations as

$$h_k = \sum_f (\sum_i W^{\boldsymbol{x}}_{if} x_i)(\sum_j t_j W^{\boldsymbol{f}}_{jf}) W^{\boldsymbol{h}}_{fk} + \sum_j t_j \boldsymbol{B}^h_{jk} \quad (20)$$

This model combines m class-dependent AEs with tied weight matrices and biases $\{\boldsymbol{W}^j, \boldsymbol{b}^j_h, \boldsymbol{b}^j_r \mid j = 1..m\}$, where each weight matrix $\boldsymbol{W}^j$ is a product of two class-independent (shared) matrices $\boldsymbol{W^x}$ and $\boldsymbol{W^h}$ and one class-dependent vector $W^{\boldsymbol{f}}_{j\cdot}$:

$$W^j_{ik} = \sum_f W^{\boldsymbol{x}}_{if} W^{\boldsymbol{f}}_{jf} W^{\boldsymbol{h}}_{fk} \quad (21)$$

The first encoder-layer ($\boldsymbol{W^x}$) learns the class-independent features, the second layer ($\boldsymbol{w^f}$) learns, how important these features are for a class and weights them accordingly. Finally, the third layer ($\boldsymbol{W^h}$) learns how to overlay the weighted features to get the hidden representation. All these layers have linear activations except for the last one. Reconstructions takes the form

$$r^j_i(\boldsymbol{x}) = \sum_k W^j_{ki} \sigma(h^j_k) + b^j_r \quad (22)$$

To learn the model, we pre-train all m autoencoders together on data across all classes, and we use the labels to determine for each observation which intermediate-level weights to train.

3.3. Performance evaluation

We tested the model (factored and plain) on the "deep learning benchmark" (Larochelle et al., 2007). Details on the data sets are listed in Table 1.

To learn the initial representations, we trained the class-specific autoencoders with contraction penalty (Rifai et al., 2011), and the factored models as denoising models (Vincent et al., 2008) (contraction

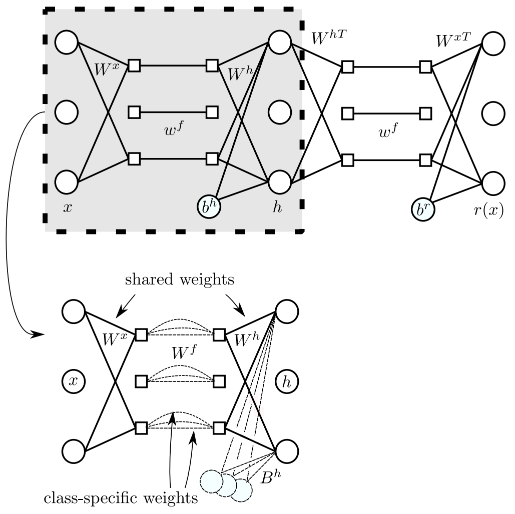

Figure 3. Factored autoencoders with weight-sharing. **Top:** a single factored autoencoder; **bottom:** encoder-part of multiple factored autoencoders that share weights. Dashed lines represent class-specific weights, solid lines represent weights that are shared across classes.

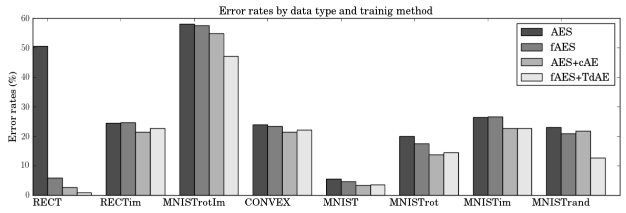

Figure 4. Error rates with and without pre-training, using ordinary and factored class-specific autoencoders.

penalties are not feasible in multilayer networks (Rifai et al., 2011)). For comparability with (Memisevic et al., 2011), we used logistic sigmoid hiddens as described in Section 3.2 unless otherwise noted. To train class-dependent autoencoders, we used labeled samples $(\boldsymbol{x}, \boldsymbol{t})$, where labels $\boldsymbol{t}$ are in one-hot encoding.

For pre-training, we fixed both the contraction penalty and corruption level to 0.5. In some cases we normalized filters after each gradient-step during pretraining. The model type (factored vs. non-factored) and parameters for classification (number of hidden units, number of factors, weight decay and learning rate) were chosen based on a validation set. In most cases we tested 100, 300 and 500 hiddens and factors. The models were trained by gradient descent for 100 epochs.

We compared our approach from Section 3 to a variety of baselines and variations:

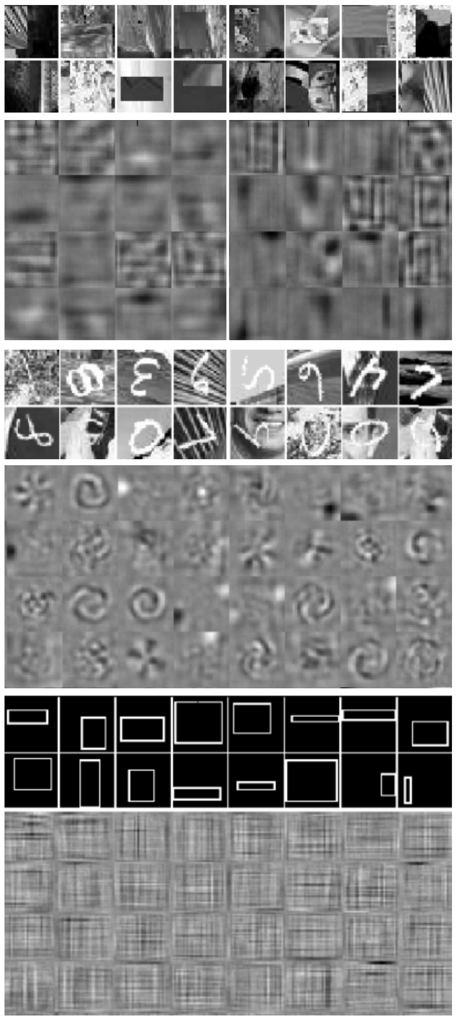

Figure 5. Example images and filters learned by class-specific autoencoders. **Top-to-bottom:** RECTimg (class 0 left, class 1 right); MNISTrotImg (factored model); RECTANGLES (factored model).

1. Train an autoencoder for each class. Then compute for each input $\boldsymbol{x}$ a vector of energies $(E_1(\boldsymbol{x}), \cdots, E_m(\boldsymbol{x}))$ (setting the unknown integration constants to zero) and train a linear classifier on labeled energy vectors instead of using the original data (Hinton, 2002).
2. Train an autoencoder for each class. Then learn only the normalizing constants by maximizing the conditional log likelihood $\sum_i \log P_{\boldsymbol{C}}(y_i|\boldsymbol{x}_i)$ on the training samples as a function of $\boldsymbol{C} = (C_1, \cdots, C_m)$
3. Optimize Eq. 19 wrt. the autoencoder parameters using back-prop, but without class-dependent pretraining.

DATA SET	TRAIN	VALID.	TEST	CL.
RECT	1100	100	50000	2
RECTIMG	11000	1000	12000	2
CONVEX	7000	1000	50000	2
MNISTSMALL	10000	2000	50000	10
MNISTROT	11000	1000	50000	10
MNISTIMG	11000	1000	50000	10
MNISTRAND	11000	1000	50000	10
MNISTROTIM	11000	1000	50000	10

Table 1. Datasets details (Larochelle et al., 2007).

DATA	SVM	RBM	DEEP	GSM	AES
	RBF		SAA$_3$		
RECT	2.15	4.71	2.14	**0.56**	0.84
RECTIMG	24.04	23.69	24.05	22.51	**21.45**
CONVEX	19.13	19.92	18.41	**17.08**	21.52
MNISTSMALL	3.03	3.94	3.46	3.70	**2.61**
MNISTROT	11.11	14.69	**10.30**	11.75	11.25
MNISTIMG	22.61	**16.15**	23.00	22.07	22.77
MNISTRAND	14.58	9.80	11.28	10.48	**9.70**
MNISTROTIM	55.18	52.21	51.93	55.16	**47.14**

Table 2. Classification error rates on the deep learning benchmark dataset. SVM and RBM results are taken from (Vincent et al., 2010), deep net and GSM results from (Memisevic et al., 2011).

4. Assign data samples to the AEs with the smallest reconstruction error.

Method 4 seems straightforward, but it did not lead to any reasonable results. Methods 1 and 2 run very fast due to the small amount of trainable parameters, but they do not show good performance either. Method 3 lead to consistently better results, but as illustrated in Fig. 4 it performs worse than the procedure from Section 3. Two lessons to learn from this are that (i) generative training of each autoencoder on its own class is crucial to achieve good performance, (ii) it is not sufficient to tweak normalizing constants, since backpropagating to the autoencoder parameters significantly improves performance.

Table 2 shows the classification error rates of the method from Section 3 in comparison to various comparable models from the literature. It shows that class-specific autoencoders can yield highly competitive classification results. For the GSM (Memisevic et al., 2011), we report the best performance of factored vs. non- factored on the test data, which may introduce a bias in favor of that model. In Tab. 3 we furthermore compare the AES performance for different activation functions on MNIST using contractive AEs with 100 hidden units. The corresponding AE energies are shown in Fig. 2. Some example images with corresponding filters learned by the ordinary and factored AES model are displayed in Fig. 5. We used the Python Theano library (Bergstra et al., 2010) for most of our experiments. An implementation of the model is available at: `www.iro.umontreal.ca/~memisevr/aescoring`

DRBM	SVM	NNET	SIGM.	LIN.	MODULUS
1.81	1.40	1.93	**1.27**	1.99	1.76

Table 3. Error rates on the standard MNIST dataset using sigmoid, linear and modulus activation functions, compared with other models without spatial knowledge.

4. Conclusion

We showed how we may assign unnormalized confidence scores to autoencoder networks by interpreting them as dynamical systems. Unlike previous approaches to computing scores, the dynamical systems perspective allows us to compute scores for various transfer functions and independently of the training criterion. We also show how multiple class-specific autoencoders can be turned into a generative classifier that yields competitive performance in difficult benchmark tasks.

While a class-independent processing hierarchy is likely to be a good model for early processing in many tasks, class-specific dynamical systems may offer an appealing view of higher level processing. Under this view, a class is represented by a dynamic *sub-network* not just a classifier weight. Such a subnetwork makes it particularly easy to model complex invariances, since it uses a lot of resources to encode the within-class variability.

Acknowledgements

We thank the members of the LISA Lab at Montreal, in particular Yoshua Bengio, for helpful discussions. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) in the project 01GQ0841 (BFNT Frankfurt).

References

Alain, G. and Bengio, Y. What regularized autoencoders learn from the data generating distribution. In *International Conference on Learning Representations (ICLR)*, 2013.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio., Y. Theano: a CPU and GPU math expression compiler. In *Python for Scientic Computing Conference (SciPy)*, 2010.

Cover, TM and Thomas, J. *Elements of Information Theory*. New York: John Wiley & Sons, Inc, 1991.

Duda, H. and Hart, P. *Pattern Classification*. John Wiley & Sons, 2001.

Gutmann, M. U. and Hyvärinen, A. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, March 2012.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, December 2005.

Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning (ICML)*, 2007.

Le, Q., Ranzato, M.A., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., and Ng, A. Building high-level features using large scale unsupervised learning. In *International Conference on Machine Learning (ICML)*, 2012.

Memisevic, R. Gradient-based learning of higher-order image features. In *the International Conference on Computer Vision (ICCV)*, 2011.

Memisevic, R., Zach, C., Hinton, G., and Pollefeys, M. Gated softmax classification. *Advances in Neural Information Processing Systems (NIPS)*, 23, 2011.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning (ICML)*, 2011.

Rolfe, J. T. and LeCun, Y. Discriminative recurrent sparse auto-encoders. In *International Conference on Learning Representations (ICLR)*, 2013.

Seung, H.S. Learning continuous attractors in recurrent networks. *Advances in neural information processing systems (NIPS)*, 10:654–660, 1998.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

Susskind, J., Memisevic, R., Hinton, G., and Pollefeys, M. Modeling the joint density of two images under a variety of transformations. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Swersky, K., Buchman, D., Marlin, B.M., and de Freitas, N. On autoencoders and score matching for energy based models. In *International Conference on Machine Learning (ICML)*, 2011.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, 2008.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

Vincent, Pascal. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, July 2011.

Welling, M., Rosen-Zvi, M., and Hinton, G. Exponential family harmoniums with an application to information retrieval. *Advances in neural information processing systems (NIPS)*, 17, 2005.

Zou, W.Y., Zhu, S., Ng, A., and Yu, K. Deep learning of invariant features via simulated fixations in video. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.