Learning Feature Representations with a Cost- Relevant Sparse Autoencoder

Martin Längkvist
School of Science and Technology, Applied Autonomous Sensor Systems
Örebro University, SE-701 82, Örebro, Sweden
Email: martin.langkvist@oru.se
http://aass.oru.se/~mlt

Amy Loutfi
School of Science and Technology, Applied Autonomous Sensor Systems
Örebro University, SE-701 82, Örebro, Sweden
Email: amy.loutfi@oru.se

There is an increasing interest in the machine learning community to automatically learn feature representations directly from the (unlabeled) data instead of using hand-designed features. The auto-encoder is one method that can be used for this purpose. However, for data sets with a high degree of noise, a large amount of the representational capacity in the auto-encoder is used to minimize the reconstruction error for these noisy inputs. This paper proposes a method that improves the feature learning process by focusing on the task relevant information in the data. This selective attention is achieved by weighting the reconstruction error and reducing the influence of noisy inputs during the learning process. The proposed model is trained on a number of publicly available image data sets and the test error rate is compared to a standard sparse auto-encoder and other methods, such as the denoising auto-encoder and contractive auto-encoder.

Keywords: sparse auto-encoder; unsupervised feature learning; weighted cost function

1. Introduction

The choice of data representation (or features) used in machine learning is known to heavily influences the performance of the method. Representation learning methods \cite{5,3,8} attempt to circumvent the need to hand-design features and instead use learning to automatically acquire representations of data. For such algorithms, the challenge is to learn relevant, robust, and invariant feature representations. The auto-encoder \cite{25,4} is one method that can be used for unsupervised feature learning. However, a shortcoming of the auto-encoder is that all inputs are treated equally and the representational capacity of the algorithm is used to minimize the reconstruction error of each input. Therefore in the presence of noisy data, an unnecessary amount of the limited representational capacity can be spent to model the noisy inputs. Furthermore, the auto-encoder is unable to distinguish between task-relevant or task-relevant information without the knowledge of the correct labels. This paper aims to improve the learning of feature representations in an auto-encoder by both reducing the influence of noise and utilizing the knowledge of the task-relevant information in the raw data by weighting the reconstruction error of every input so that the learned features are more focused on the task-relevant information in the input data.

Several strategies proposed in the literature for guiding the learning of feature representations have been proposed in the literature. For example, sparse feature representations are achieved by adding a sparsity constraint \cite{19,20}. Robustness is achieved in
a contractive auto-encoder by penalizing changes in the feature representation for small changes in the input and in a denoising auto-encoder by reconstructing clean input from corrupted input. The use of noise injection in the training examples has also been done to improve generalization.

The use of dropout also guides the features to learn meaningful representations independently and reduce the co-adaptation between the hidden units by randomly dropping half of the hidden units for each training example. While these methods show promise for guiding the learning of features, they mainly focus on the unsupervised setting where the distinction between task-relevant and task-irrelevant patterns is still a challenge.

A method that attempts to solve the problem of finding task-relevant patterns from scratch with the help of labels is a modification of the RBM named the point-wise gated Boltzmann machine (PGBM). The PGBM uses a feature selection algorithm to divide the learned features from a pre-trained RBM into a group of task-relevant features and a group of task-irrelevant features. A binary switching unit for each input pixel is used to determine what pixels are task-relevant. The classification is only performed on the task-relevant group of features. The disadvantage of the PGBM is that it requires an pre-trained network before evaluating the inputs and the learned features and still spends half of the model's capacity on modeling the task-irrelevant patterns. The proposed method in this work aims to make every learned feature task-relevant without the need for pre-training.

The proposed method is also related to cost-sensitive learning. Cost-sensitive learning is used during supervised learning and addresses the issue that sometimes the classification error for each category should not be treated as equal. For example in medical diagnosis, where a false-negative should be considered more costly than a false-positive because the former could result in a loss of a life. Cost-sensitive learning introduces a cost matrix, \(C(i,j)\) that weights the cost of classifying the predicted class \(j\) as actual class \(i\). The cost for a correct classification is \(C(i,i) = 0\) and if the costs are equal the off-diagonal values are \(C(i,j) = 1\). The main difference between cost-sensitive learning and our method is that our method weights the reconstruction error of each input unit during the unsupervised phase of learning the auto-encoder instead of weighting the classification error of each category during supervised learning.

The proposed model presented in this work is designed for data sets that contain many factors of variations and reduces the cost for reconstruction error for irrelevant inputs. The weighting of reconstruction error has also been done by adding a penalty term for the squared error cost between an optimal hidden unit activation and the actual hidden unit activation. The added penalty term has the effect of reducing the allowed parameter space by training the model to have a high reconstruction error for inactive inputs. The optimal hidden unit activation is obtained by fixing the model parameters and then the model parameters is updated with the obtained optimal hidden layer activation. Finally, weighting the error term has also been done to avoid poor local minima by preventing early saturation of the hidden units.

The goal of the proposed method is to learn more meaningful feature representations from data that has a high amount of irrelevant patterns. The method is trained on a number of visual recognition benchmark data sets and evaluated by examining the learned features and comparing the test error rates to other unsupervised feature learning algorithms.

This paper is organized as follows. Section 2 gives an introduction to the sparse auto-encoder. The proposed model is presented in Section 3. The experiments are presented in Section 4 and finally a conclusion and future directions is given in Section 5.

### 2. Sparse Auto-Encoder

The model that is used in this work is the auto-encoder. The auto-encoder consists of one input layer, one or more hidden layers, and one output layer. The goal of an auto-encoder is to reconstruct the input data via an encoder and a decoder. The feed-forward activations in the encoder from the visible units \(v_i\) to the hidden units \(h_j\) is expressed as:

\[
h_j = \sigma_f \left( \sum_i W_{ji} v_i + b_j \right)
\]

where \(\sigma_f(.)\) is the activation function for the encoder. In this work the sigmoid function \(\sigma_f(x) = \frac{1}{1+e^{-x}}\) is used for the encoder activation function. In the decoder phase, the last hidden layer is decoded back to reconstructions of previous input layers. One pass of
an decoder with tied weights is calculated as:

\[
\hat{v}_i = \sigma_g \left( \sum_j W_{ij} h_j + b_i \right)
\]  

(2)

The activation function in the last decoder depends on the nature of the input in the first visible layer. For a visible layer with assumed Gaussian distribution, a linear activation function \( \sigma_g(x) = x \) can be used.

For a probabilistic interpretation of the auto-encoder, the cost function is defined as the negative log likelihood of the probability distribution of the visible layer given the hidden layer. If we assume this distribution has a Gaussian distribution with mean equal to the reconstruction of the visible layer, \( \hat{v} \), and identity covariance, i.e., \( P(v|h) = \mathcal{N}(\hat{v}, I) \) we get:

\[
L = -\log P(v|h) = \frac{1}{2N} \sum_{n=1}^{N} \sum_i (v_i^n - \hat{v}_i^n)^2
\]  

(3)

which aims to minimize the average squared reconstruction error of a training batch of \( N \) training examples.

A regularized auto-encoder restricts the allowed parameter space and is obtained by adding regularization terms. A common choice is to add a L2 weight decay term, \( \frac{1}{2} \sum_i \sum_j (W^{(l)}_{ij})^2 \), that keeps the weight matrices in each layer \( l \) small, and a sparsity penalty term, \( \beta \sum_j KL(p_i | p_j) \), where \( KL(p_i | p_j) = p_i \log \frac{p_i}{p_j} + (1-p_i) \log \frac{1-p_i}{1-p_j} \) is the Kullback-Leibler (KL) divergence and \( p_j \) is the mean activation for hidden unit \( j \) over all training examples in the current minibatch. The inclusion of these regularization terms prevents the trivial learning of a 1-to-1 mapping of the input. Each regularization term comes with one or more hyper parameters (\( \lambda, \beta, \rho \)) and can be set with a full grid search, random grid search \(^2\), or hyperparameter optimization \(^6\).

3. Cost-Relevant Sparse Auto-Encoder

An issue with training sparse auto-encoders using the cost function in Eq. 3 is that the reconstruction error of each input unit is treated equal. This comes from the assumption that each input unit given the hidden units has a Gaussian distribution with unit variance. If we instead set the precision (inverse variance) as a learnable parameter, i.e., \( P(v|h) = \mathcal{N}(\hat{v}, \alpha^{-1}) \) we get the following cost function:

\[
L = -\log P(v|h) = \frac{1}{2N} \sum_{n=1}^{N} \sum_i (v_i^n - \hat{v}_i^n)^2 \alpha_i - \sum_i \log \alpha_i
\]  

(4)

The changes from a regular sparse auto-encoder are the element-wise multiplication of the reconstruction error for the \( i \)th unit with the weighting vector \( \alpha \) and the added weighting vector penalty term. If \( \alpha_i = 1 \) for \( \forall i \) the model generalizes to a regular sparse auto-encoder. If \( \alpha_i \) is small than the reconstruction error for the \( i \)th unit is decreased and the amount by which the weights and biases in the previous layers are changed during back-propagation is decreased compared to if \( \alpha_i = 1 \). Similarly, for \( \alpha_i > 1 \) the reconstruction error and the size of parameter update is increased. Therefore, the weighting vector \( \alpha \) determines which units should be focused on and have a larger impact on the parameter updates and which input units should have a lesser impact.

A crucial aspect of the proposed model is how to determine the values of the weighting vector. In this work we explore two methods, namely using a feature selection method to pre-calculated the values (fixed weighting vector), as well as learning the values together with the model parameters (adaptive weighting vector). The fixed weighting vector method is designed as a supervised method where the knowledge of the correct labels is utilized and the adaptive weighting vector method is purely unsupervised.

3.1. Fixed weighting vector

In this setting, \( \alpha_i \) is set to a fixed value for each input unit \( i \). Since the values of \( \alpha_i \) do not change, the weighting vector penalty term in Eq. 4 is removed. The values can be calculated in a number of ways depending on the data set or if prior knowledge about the data can be utilized. In this work, a feature selection algorithm, namely the score from the t-test algorithm on the input data, is used to set the values of the weighting vector. The scores are normalized to values between 0 and 1 by:

\[
\alpha_{\text{norm}} = \frac{\alpha - \min \alpha}{\max \alpha - \min \alpha}
\]  

(5)

and then normalized so that \( \sum_i \alpha_i = d \), where \( d \) is the number of input units. This reduces the difference between the reconstruction error and the other regularization terms so that the hyperparameters do not need to be re-adjusted.
Furthermore, since this method requires the knowledge of the correct labels, a separate weighting vector $\alpha^{(k)}_i$ is used for each category $k$. This allows the model to change focus depending on the category that the current input data belongs to.

### 3.2. Adaptive weighting vector

When using an adaptive weighting vector, the values of the weighting vector are learned together with the model parameters. The weighting vector is parametrized according to $\alpha = \log(1+\exp(a))$, where $a$ is the learnable parameter. This parametrization is done in order to avoid numerical problems, i.e., keeping $\alpha > 0$. If $\alpha_i$ decreases from 1, the square error term will decrease and the weighting vector penalty term will increase. However, for values of $\alpha_i$ above 1 the reconstruction error will increase but the weighting vector penalty term is negative. A hyperparameter $\gamma$ is introduced to the weighting vector penalty term in Eq.4 in order to control the trade-off between reconstruction error and the weighting vector values deviating from 1 and prevent the values to blow up. During learning, the algorithm determines a weighting vector that minimizes the total cost function.

### 4. Experiments

This section is divided into two parts where the experiments are performed with a fixed or adaptive weighting vector. Early-stopping is used in all experiments to prevent overfitting in both pre-training and fine-tuning. Training is stopped if the accuracy for a held-out validation set is not improved in the last 20 epochs. Mini-batch stochastic gradient descent with decaying learning rate and momentum is used as optimization method.

The hyperparameters are set with random grid-search and the combination of hyperparameters that gives the best accuracy on the validation set is used. The weight decay $\lambda$ ranges between $10^{-3}$ to $10^{-5}$, sparsity parameter $\beta$ ranges between 3 and 0.003, weighting vector penalty parameter $\gamma$ ranges between 1 and 0.01 (for fixed weighting vector this parameter is not used), and learning rate ranges between 0.1 and 0.001. The number of hidden units is set to 500, which is a relatively small size compared to other approaches. Therefore, our aim is to achieve comparable results but with a smaller network since the capacity of the model is not spent on irrelevant patterns.

#### 4.1. Fixed weighting vector

The use of a fixed weighting vector is evaluated using two data sets, namely MNIST variations and Jittered-Cluttered NORB.

##### 4.1.1. MNIST variations

The MNIST variations data set consists of 4 variations of the standard MNIST handwritten numbers: added background noise (mnist-back-nmd), added background images (mnist-back-images), rotated numbers (mnist-rot), and rotated numbers with background images (mnist-rot-back-images). Each data set consists of 10000 training examples, 2000 validation examples, and 50000 test examples. The input size is $28 \times 28$. The input data is seen in Fig. 2(a).

Fig. 1 shows the values of the weighting vector for all four data sets when it is calculated using the t-test feature selection algorithm on the input data. Each column is one of the four data sets. Top row show the mean of the scores over all classes and the bottom row show the raw scores for each class. The use of a separate weighting vector for each class further indicates which input units should be focused on for each class. The first two data sets assign a focus for the center of the image while the last two data sets focus on rings with different widths and radii. The white sections in the second row indicate that high pixels intensities in the input data play a role for that class and the black parts indicate that the absence of a pixel intensity is significant. Gray sections indicate class-insignificant input pixels. The weighting vector for each data set and class is set to the maximum score of the t-test algorithm for each category. The values are normalized according to Eq. 5.
Learning Representations with a Cost-Sensitive Sparse Autoencoder

Fig. 1. Raw scores from the t-test algorithm on the training images. Each column from left to right is the data sets mnist-back-images, mnist-back-rand, mnist-rot, and mnist-rot-back-images. Top row show the mean of the t-test score over all classes. Bottom row show the scores for each class. Values are plotted with switched sign for a more intuitive visualization.

A cost-relevant sparse autoencoder with fixed weighting vector is trained on the input data. The number of hidden units is set to 500 and the other hyperparameters were set as mentioned in the beginning of this section. Fig. 2 shows the reconstructions and learned features of a regular autoencoder and the cost-relevant autoencoder trained on mnist-back-image. The reconstructions from the cost-relevant autoencoder resemble a depth-of-field effect where the background is more blurred out compared to the reconstructions of a regular autoencoder. The learned features of the cost-relevant autoencoder capture little information from the outer part of the image and instead focuses on pinstrokes.

The test error rate for a sparse autoencoder (SAE) and a cost-relevant sparse autoencoder (CR-SAE) with a fixed weighting vector is presented and compared to previous works in Table 1a. The goal of our approach is to achieve comparable results to other models while using a single layer with a small number of hidden units. Our model uses 500 hidden units while the other works use model sizes between 500 and up to 3000 (6000 for SAA-3) per layer. The CR-SAE improved the classification result for all data sets compared to a SAE, NNet, and SVM (except for mnist-rot). For mnist-back-images the CR-SAE also achieved a lower classification error than a 3-layered sparse auto-associator. For mnist-back-mind comparable results to SdA-3 and CAE-2 were achieved and a lower error than SAA-3, and CAE-1. For mnist-rot CR-SAE also performed better than DBN-1 but not better than any of the deep networks or CAE and this is probably because mnist-rot does not contain any added noise so there is little gain in paying less attention to a subset of the input. For mnist-rot-back-images a lower classification error is also achieved compared to SAA-3, DBN-1 and CAE-1.

The supervised PGBM and CAE-2 achieved a lower test error rate on all 4 data sets than the proposed method. The CAE-2 uses two layers of hidden units with 1000 units in each layer while our method only uses one layer with 500 hidden units. One possible reason for the lower test error for the supervised PGBM is that it uses generative feature selection both at the learned features from a pre-trained RBM (high-level) and at the visible units by blocking task-irrelevant units (low-level) while the proposed method of fixed weighting vector only uses feature selection on the low-level input units before training.

4.1.2. Jittered-Cluttered NORB

The Jittered-Cluttered NORB data set consists of images from the NORB data set where the central object has been jittered and disruptive background clutter and a distractor object has been added in order to produce a more challenging data set. The images are of five small toy figures (animal, human figure, airplane, truck, and car) taken at 6 lighting conditions, 9 elevations, and 18 azimuths using a stereoscopic camera. The input size is $108 \times 108 \times 2$. Some examples can be seen in Fig. 4(a). There is also a sixth category with no toy figure present in the image and only background clutter and a distractor object.

The validation set is created by randomly drawing 19.4% of the images from the training set which resulted in a training set of 234446 images, a valida-

---

For the recalculated results from some of the models see: http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007
Fig. 2. (a) Input data, (b) reconstruction of regular auto-encoder, (c) learned features for regular auto-encoder, (d) reconstruction of cost-relevant auto-encoder with fixed weighting vector, and (e) learned features for cost-relevant auto-encoder with fixed weighting vector. Each row from top to bottom is the data set \textit{mnist-back-images}, \textit{mnist-back-rand}, \textit{mnist-rot}, and \textit{mnist-rot-back-images}.

tion set of 57154 images and a test set of 58320 images. The sets are pre-processed in the same way as a previous work\cite{22}, that is, each image is downsampled to $32 \times 32 \times 2$ and normalized by subtracting the mean of the image and divided by the average standard deviation of all training pixels.

The training set is used to train a sparse auto-encoder (SAE) using 500 hidden units and a cost-relevant learning sparse auto-encoder (CR-SA E) with fixed weighting vector using 500 and 2000 hidden units. The same hyperparameter search and optimization method previously used in Section 4.1.1 are also applied here. The networks were first pre-trained and then fine-tuned with a softmax layer attached on top of the hidden layer. Due to the size and nature of the training set, the values of the weighting vector are updated with the t-test algorithm for each training mini-batch. Fig. 3 shows the values of the weighting vector for three different training batches.
Table 1. Classification errors [%] with 95% confidence intervals on MNIST variations using cost-relevant sparse auto-encoder (CR-SAE) with fixed weighting vector, sparse auto-encoder (SAE), supervised neural net (NNet), SVM with Gaussian kernel, 3-layered stacked auto-associator (SAA-3), 1 and 2-layered contractive auto-encoder (CAE-1, CAE-2), 3-layered stacked denoising auto-encoder (SdA-3), deep belief network (DBN-1), and a supervised point-wise gated Boltzmann machine (supervised PGBM).

<table>
<thead>
<tr>
<th>Model</th>
<th>mnist-back-images</th>
<th>mnist-back-rand</th>
<th>mnist-back-rot</th>
<th>mnist-back-rot-back-images</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR-SAE</td>
<td>18.59 ± 0.35</td>
<td>10.98 ± 0.28</td>
<td>12.86 ± 0.30</td>
<td>47.68 ± 0.45</td>
</tr>
<tr>
<td>SAE</td>
<td>19.97 ± 0.36</td>
<td>13.60 ± 0.31</td>
<td>14.23 ± 0.38</td>
<td>53.56 ± 0.45</td>
</tr>
<tr>
<td>NNet 17</td>
<td>27.41 ± 0.39</td>
<td>20.04 ± 0.35</td>
<td>18.11 ± 0.34</td>
<td>62.16 ± 0.43</td>
</tr>
<tr>
<td>SVM (r_{bf}) 17</td>
<td>22.61 ± 0.37</td>
<td>14.58 ± 0.31</td>
<td>11.11 ± 0.28</td>
<td>55.18 ± 0.44</td>
</tr>
<tr>
<td>SAA-3 17</td>
<td>23.00 ± 0.37</td>
<td>11.28 ± 0.28</td>
<td>10.30 ± 0.27</td>
<td>51.93 ± 0.44</td>
</tr>
<tr>
<td>CAE-1 27</td>
<td>16.70 ± 0.33</td>
<td>13.57 ± 0.30</td>
<td>11.59 ± 0.28</td>
<td>48.10 ± 0.44</td>
</tr>
<tr>
<td>CAE-2 27</td>
<td>15.50 ± 0.32</td>
<td>10.90 ± 0.27</td>
<td>9.66 ± 0.26</td>
<td>45.23 ± 0.44</td>
</tr>
<tr>
<td>SdA-3 30</td>
<td>16.68 ± 0.33</td>
<td>10.38 ± 0.27</td>
<td>10.29 ± 0.27</td>
<td>44.49 ± 0.44</td>
</tr>
<tr>
<td>DBN-1 17</td>
<td>16.15 ± 0.32</td>
<td>9.80 ± 0.26</td>
<td>14.69 ± 0.31</td>
<td>52.21 ± 0.44</td>
</tr>
<tr>
<td>supervised PGBM 28</td>
<td>12.85 ± 0.30</td>
<td>6.87 ± 0.23</td>
<td>–</td>
<td>44.67 ± 0.44</td>
</tr>
</tbody>
</table>

Table 2. Classification errors on the Jittered-Cluttered NORB data set with the proposed method and previous works.

Fig. 3. Fixed weighting vector for each class of animal, human, airplane, truck, car, and none (columns) for three training mini-batches (rows).

Fig. 4 shows the input data, reconstruction, and some of the learned features in no particular order. The values of the weighting vector for the "none" category are mostly high around the edges while the airplane, car, and truck category mostly has a high focus in the middle. The animal and figure category has a more scattered distribution of the weighting vector between each training mini-batch.

Table 2 shows the test error rate for our models and previous works. The CR-SAE with 500 hidden units achieves a lower test error rate compared to a SAE but not lower than a RBM with 4000 hidden units or a multi-layered convolutional net. When the number of hidden units is increased to 2000, the CR-SAE achieved a lower test error rate than a RBM with twice the number of hidden units with rectified linear units and a 6-layered convolutional net.

Table 3. Confusion matrix [%] of Jittered-Cluttered NORB from a cost-relevant sparse auto-encoder with 2000 hidden units.

4.2. Adaptive weighting vector

The use of an adaptive weighting vector can be used instead of a fixed weighting vector when the training data is unlabeled and there is no obvious way of setting the weighting vector. Another advantage of using an adaptive weighting vector is when the quality of a subset of the input changes over time. The use of an adaptive weighting vector is tested by comparing the classification accuracy on the small NORB data set with partially corrupted input using a sparse auto-encoder with adaptive weighting vector and a regular sparse auto-encoder.
Fig. 4. (a) Input data for Jittered-Cluttered NORB where each row is one category, (b) reconstruction of the input data with a CR-SAE, and (c) a subset of the learned features of CR-SAE.

4.2.1. Small NORB

The Small NORB dataset consists of images of size $96 \times 96$ of 5 categories (animal, human figure, airplane, truck, and car) taken at 6 lighting conditions, 9 elevations, and 18 azimuths. The training and testing set consists of 24300 images each. The images are downsampled to $48 \times 48$ and normalized with local contrast normalization with patch-size $9 \times 9$ resulting in images of sizes $40 \times 40$. The $25 \times 25$ center patch is selected to further reduce the dimensionality.

The training data is locally corrupted with Gaussian noise with standard deviation 0.1 or 1 with a patch of varying size. The location of the patch is set randomly and is the same for all training images. The inputs outside the corruption patch are unaffected. Fig. 5 shows the classification accuracy on the small NORB data set with varying corruption patch size. When no noise is added to the training data, the test accuracy rate using a CR-SAE with adaptive weighting vector is similar to a SAE. For small amounts of noise (standard deviation $= 0.1$), the difference in the classification accuracy between the two models is unnoticeable. When the training data is effected with a large amount of noise (standard deviation $= 1$) the classification accuracy using a CR-SAE with adaptive weighting vector is significantly higher than a SAE.

In the next experiment a moving corruption patch of size $8 \times 8$ or $16 \times 16$ is added to the training images to examine the how the weighting vector adapts over time. The noise patch is moving randomly one pixel after each training example. A CR-SAE with adaptive weighting vector and a SAE is trained on the clean small NORB dataset and with the added local noise. Fig. 6 shows the corrupted NORB input data at different times and the values of the weighting vector.

Fig. 7 shows the classification accuracy for the moving corruption patch. The difference in the accuracy between the two models is small for clean input data (0-by-0 noise patch). The classification accuracy is higher with a CR-SAE with adaptive weighting vector when a medium-sized $8 \times 8$ patch is corrupting the input data - and the difference is even
more noticeable for a noise patch of size 16 × 16. The classification accuracy did not improve when using an adaptive weighting vector on noise-free input data.

4.2.2. MNIST variations

The method of using an adaptive weighting vector is tested on the four MNIST variations data sets. Fig. 8 shows the final values of the weighting vector. For mnist-backg-images (top-left), the weighting vector assigns low values to the background and focuses on the center area where the digit is located and has a more varying input. For mnist-backg-rand (top-right), the opposite effect is achieved since the digit is much easier to reconstruct than random noise. For the rotating digit, the focus becomes a ball-shaped area at the center of the image.

5. Conclusion

This paper has shown that category-specific selective attention can be achieved by weighting the reconstruction error and improve the learning of feature representations in the unsupervised pre-training phase for data sets with irrelevant patterns. The method achieves a lower test error rate compared to a standard sparse auto-encoder and achieves comparable results to other methods that use larger models. A feature selection technique that utilizes the knowledge of the correct labels has been explored.
for setting the selective input and an unsupervised method for automatically learning the selective input has been proposed.

Future work include further exploring of methods for setting the weighting vector without using the knowledge of the labels. The possibility to combine the use of a weighting vector with other feature learning techniques and the usefulness of using the method with a RBM or domains such as convolutional settings and time-series data is another interesting future direction.

Acknowledgements

We greatly appreciate the anonymous reviewers for their valuable comments that helped in improving the quality of this paper.

References

5. Y. Bengio and Y. LeCun, Scaling learning algorithms towards AI, Large-Scale Kernel Machines, eds. L. Bottou, O. Chapelle, D. DeCoste and J. Weston (MIT Press, 2007).


