Training Neural Networks with Policy Gradient

Conference Paper · May 2017

DOI: 10.1109/IJCNN.2017.7966360

2 authors, including:

Sourabh Bose
University of Texas at Arlington
4 PUBLICATIONS **5** CITATIONS

SEE PROFILE

Training Neural Networks with Policy Gradient

Sourabh Bose and Manfred Huber
Computer Science and Engineering
University of Texas at Arlington
Arlington, TX USA
Email: sourabh.bose@mavs.uta.edu huber@cse.uta.edu

***Abstract*—Neural networks are a powerful function approximation tool which has the ability to model any function with arbitrary precision. For any function as a black box, it is able to reconstruct the function given the target and the input data. However, there are problems where the target is at least partially unknown. In such cases it is impossible for a traditional neural network to compute the gradient of the system. This problem is evident in sparse autoencoder systems where lateral inhibitions are required. Lateral inhibitions are usually imposed with extra lateral connections among nodes in the immediate vicinity of the hidden layer. However, it is computationally expensive, and results in a very complex architecture which is not scalable to real world problems. Thus it is often necessary to achieve structural constraints without such complexity limitations. Similar situation arises in case of multi-label classification problems without a known target. In such cases only an evaluation is accessible to the network, which states whether the sample was correctly classified or not. In this problem, it is necessary to derive the gradient for this partial information to solve the problem.**

The proposed model solves such problems using policy gradient algorithms. This approach allows for imposing any arbitrary non-differentiable constraints on the neural network system by deriving the required gradient from a system of rewards. Furthermore a novel form of sparsity with lateral inhibitions over the entire hidden layer is proposed. It is shown that the proposed model is not only able to achieve a much tighter and sparser representation, it also achieves similar or better results than traditional forms of sparsity.

I. Introduction

A neural network is a powerful modeling tool which has the ability to approximate arbitrarily complex functions, provided the optimization function has a gradient. However, it falls short if the function to be optimized does not have an inherent gradient or if the function to be optimized is a black box where the equation is unknown.

There are many problems where such scenarios are evident. For example, In a classification task where instead of the target, a function outputs if the classification was correct or not. Traditional classification neural networks fail to work where instead of a target is not specified,.

A similar problem arises where lateral inhibitions on a sparse autoencoder are to be imposed on the features. Imposing such constraints, usually termed as structural constraints, is difficult since they require lateral connections among the nodes in a particular layer. This results in a complicated, recurrent network architecture requiring gradient updates for a node to be affected by sibling nodes connected in a distributed fashion. However, imposing such a constraint without lateral connections among the hidden nodes is not possible as the gradient is non-computable without the error and activation information from the neighbor nodes.

The proposed method solves such problems where the gradient is non-differentiable or simply non-computable by modeling an approximate smooth representation of the optimization function, and subsequently deriving the gradient from that approximation instead.

A similar gradient problem is present in case of solving reinforcement learning problems with neural networks, where there is no inherent target but only a system of rewards which is a qualitative utility of the output. In this case a secondary critic network is used to approximate the gradient of the system of rewards with respect to the output actions at each state. This method is called policy gradient where one network learns which actions to take while another network approximates the required gradient of the rewards with respect to the output actions[1].

The proposed method uses a modification of the policy gradient algorithm and represents the problem as a context based n-armed bandit problem in order to enforce the non-differentiable or non-computable constraints from a system of rewards. The functions are considered to be a black box, modeled in a critic network where the state is the input sample and the actions are the activations from nodes on which the constraint is to be applied. The constraints are represented as a reward function which is modeled by the critic network. Given any reward function which might be non-differentiable the critic network learns a smooth approximation of the constraint. The gradient of the output of this network with respect to the activations is then fed into the actual actor network.

Experiments carried out show that the proposed method is able to solve problems which impose arbitrary structural constraints on the networks in addition to minimizing the reconstruction error. Additionally it is shown that it is able to solve multi-label classification problems without an actual target but with a correct or wrong outcome indicator. Finally, a novel form of structural lateral sparsity is introduced which results in much sparser representations without degrading the reconstruction error. Lateral inhibitions are shown to learn more concise and an order of magnitude more sparser features with better generalizations compared to traditional KL sparsity[2]. However, this form of sparsity is rarely used in the real world due to their high computational complexity and complex architecture design. The proposed approach negates

the need for interconnected nodes in the hidden layer for lateral inhibitions, thus reducing the complexity, while learning better laterally sparse features, thus facilitating their use in real world applications which was not possible earlier.

II. Existing Methodologies

Non-differentiable constraints are usually optimized by converting the function into a multitude of convex piecewise linear functions or solving the function with a hill climbing algorithm like simulated annealing. The function is then solved based on the piece the function currently lies on [3]. However, this is not possible for functions like the L0 norm which do not have a gradient for any piece. On the other hand solving such constraints using simulated annealing is infeasible for higher degrees of freedom of the constraint.

Another approach to solving such problems is to consider one variable at a time of the constraint and solve for it using sequential minimal optimization (SMO)[4]. The constraint is formed as a dual problem, for example using Moreau-Yosida regularization [5], for the SMO algorithm.

Existing methods of imposing a sparsity constraint involve forcing a feature to fire as few times as possible for all samples [6]. KL-divergence between the current average activation and a given target average activation is used to compute the gradient for the sparsity constraint. However, another type of sparsity in the form of lateral inhibitions is often imposed. This form of sparsity tries to minimize the number of active features required to reconstruct a sample.

Existing methods achieve laterally inhibited sparse features using lateral connections among nodes in a hidden layer [2]. Such connections are needed in order to share information about which nodes have fired in the locality. The gradient for a feature node is computed with respect to the laterally connected nodes. Although resulting in a tighter and more compact representation of the data, it is computationally expensive and requires a complex network architecture. Thus, only a few neighbor nodes are locally connected in order to reduce the complexity while retaining some lateral sparsity information.

III. overview of Reinforcement learning,actor critic and policy gradient

Reinforcement learning is a class of algorithms which learn decision making systems[7]. Given a state, which represents the current status of the problem, an agent needs to learn which action to take in order to maximize a reward function. However, the agent does not know the correct action for each state and has to learn this from intermediate or delayed rewards possibly at the end of the entire sequence of actions. The agent learns the best possible action to take by randomly exploring all actions for the states and computes a utility value for each state action pair. This utility function is known as the value function. This utility value is then used to determine the best action to take at each state.

There are many Reinforcement learning algorithms which are used to learn this utility value or Q-value. The SARSA algorithm [8] updates a Q-value for a state action pair as a function of the reward for the current state action pair, and the utility of the next state and action to be taken [8]. This is known as the Bellman equation.

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha(R(S_t, a_t) + \gamma(Q(S_{t+1}, a_{t+1})) - Q(S_t, a_t)) \quad (1)$$

To solve such a system, which inherently does not have a gradient towards the best possible action, using neural networks, the policy gradient algorithm is used [9] [10]. This method, uses an actor-critic model, where the actor learns the best action to take as an output for a given input state [11]. The actor network learns a probability distribution over the set of actions for a given state. An action is chosen based on this distribution [12]. The critic network learns the value function with the current state and chosen action as input. The gradient of the critic network value output with respect to the actions is then used to train the actor network [13].

However, there are some simpler problems where there is no next state. In such cases, taking an action from the state results in termination and a reward. The agent has to learn which action results in the highest reward. Such systems are usually known as n-armed bandit systems [14].

Additionally, there are problems where there exist multiple states, but taking an action in any state results in a termination and a reward. In such cases the agent, similar to an n-armed bandit, has to learn which action results in the maximum reward for every state in the system. This is a slightly different problem than n-armed bandit, and is usually known as a context-based n-armed bandit system [15].

IV. Approach

The proposed approach uses policy gradient to impose constraints on the network instead of learning actions. It solves the problem of constraint satisfaction using an actor critic model. The given constraint C, is encoded as a system of rewards $R(S_i, a_c)$ which is a metric representing the satisfaction of the constraint. The critic network approximates this reward function and provides the gradient with respect to the activations to the actor network. In constrast to policy gradient where the input to the critic network is the action taken by the actor network, the network models the reward function from the sample and activations of the nodes where the constraint is to be applied.

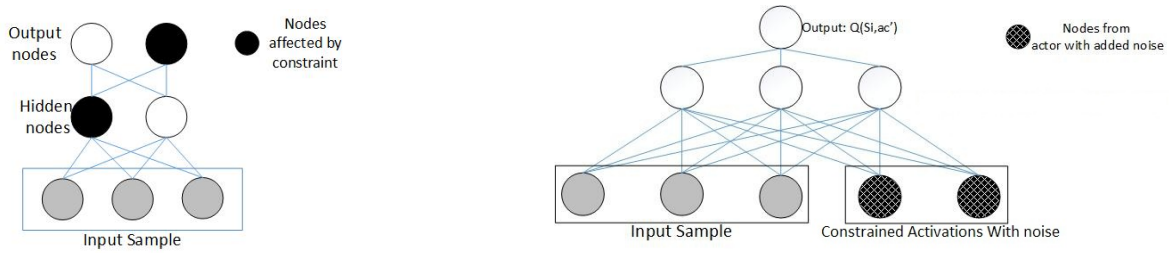
Let A be the set of activations in the actor network for all nodes in the input layer a_i, hidden layer activations a_h and the output layer activations a_o.

$$A = \{\{a_i\}, \{a_h\}, \{a_o\}\}$$

For a given sample S_i, a_c is defined as the activation of the set of nodes affected by the given constraint C over the set of all activations A in the actor network as shown in Figure 1(a).

$$a_c \subseteq A$$

For example, in case of the classification problem a_c is the set of all activations in the output layer, a_o. In genral, however,



(a) Actor network

(b) Critic network

Fig. 1: General network architecture. (a) Actor network, shaded grey nodes indicate input nodes while black nodes specify the nodes over which the constraints are to be applied. (b) Critic network, checkered black nodes indicate values from constrained nodes with added noise. Shaded grey inputs and activations of affected nodes used as input to critic network, the output being the utility for the activations produced

the activations where the constraint is to be applied can be a combination of a_h and a_o activations from different layers.

Similar to policy gradient systems, in order to train the critic network, some exploration is required to approximate the space defined by the reward function. In order to achieve this, a small zero-mean gaussian noise is added to the activation outputs of the affected nodes. This process is similar to exploring actions in policy gradient and allows the system to explore new nearby activation values which might lead to better rewards during training.

$$a'_c = a_c + \mathcal{N}(0, \sigma^2)$$

The new activations a'_c so formed, in addition to the given sample is then used as an input to the critic network as shown in Figure 1(b). Given the reward from the constraint C the critic network then approximates the utility of the activations for the given sample.

For each sample, the activations a'_c with the added noise, in the actor network are used as inputs to the critic network along with the input values for the sample. The critic network is trained with the reward from the constraint as the target. This reward might be a one or zero as in the case of the classification problem without a target dataset or in the case of sparsity with lateral inhibitions, the reward might be the L0 norm of the constraint nodes per sample.

The proposed model represents the problem as a context based n-armed bandit system, where there is no next state, and for each sample there exists a set of activations which maximizes the reward. The target of the critic network drifts during training and depends on the current output of the network. The target value of the network is given by the Bellman equation. The utility $Q(S_t, a'_{c_t})$ of a set of activations a'_{c_t} for a given sample S_t, ignoring the next state is computed as

$$Q(S_t, a'_{c_t}) = Q(S_t, a'_{c_t}) + \alpha(R(S_t, a'_{c_t}) - Q(S_t, a'_{c_t})) \quad (2)$$

The critic network is trained by minimizing the mean-squared error of the output of the critic network and the target. The target of the network maintains an adaptive average and over time converges to the expected reward $E[R(S_t, a'_{c_t})]$, for activations a'_{c_t} and sample S_t. Given the error function as the mean-squared error where the output y is given as the current utility learned by the network for sample S_t and activation values a'_c

$$y = Q(S_t, a'_{c_t}),$$

the new target τ computed as a function of the current output y and the reward incurred is given as

$$\tau = Q(S_t, a'_{c_t}) + \alpha(R(S_t, a'_{c_t}) - Q(S_t, a'_{c_t}))$$

The mean squared error to train the critic network is computed as

$$Error = \frac{1}{2}(\tau - y)^2$$

$$= \frac{1}{2}(R(S_t, a'_{c_t}) - Q(S_t, a'_{c_t}))^2$$

The gradient of the critic network is computed as

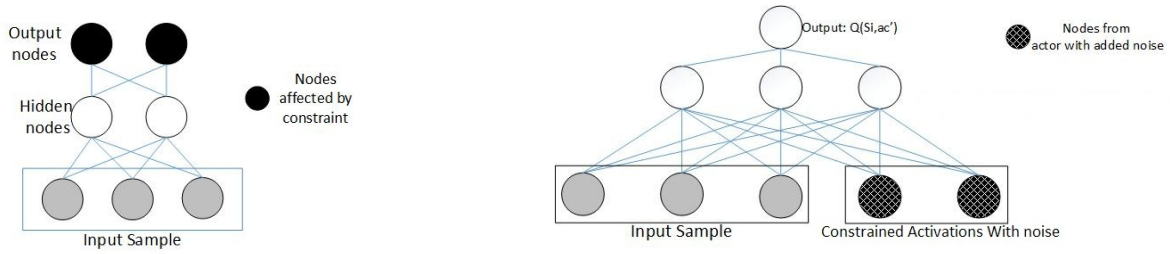
$$\frac{\partial Error}{\partial Q(S_t, a'_{c_t})} = R(S_t, a'_{c_t}) - Q(S_t, a'_{c_t})$$

which is the Bellman error for n-armed bandit systems[16]. This gradient is then backpropagated to train the critic network. The weight w update is then computed as

$$\frac{\partial Error}{\partial w} = \frac{\partial Error}{\partial y}\frac{\partial y}{\partial w}$$

The gradient for the layers is backpropagated for the entire network while training. The only difference is that the target is drifting over time and the gradient is the Bellman error.

The critic network is trained using this gradient and therefore tries to approximate the reward function and output as closely as possible. If the reward function is non-differentiable, it learns a smooth approximation for the function. On the other hand, the actor network tries to maximize the reward function. This is a different gradient than the one used to train the critic network which in contrast tries to get consistent reward function outputs. To obtain the constraint related error for the actor network, a separate gradient is computed during backpropagation in the critic network which is the gradient



(a) Actor network

(b) Critic network

Fig. 2: Target-less classification network. (a) Actor network, shaded grey nodes indicate input nodes while constraints are applied on the output layer nodes. (b) Critic network, with the input and noisy actor output node activations as input

without the bellman error. This function is the gradient of the output y of the critic network which is $Q(S_t, a'_{c_t})$ with respect to the activations a'_c.

$$\frac{\partial Q(S_t, a'_{c_t})}{\partial a'_{c_t}} \tag{3}$$

Analogous to the policy gradient system, the critic network learns the value function whose output is defined by the rewards from the constraint system. The probabilistic actions are replaced by the noisy activations of the actor network.

Thus, in order to train the system, the actor network is forward propagated to record the activations for the nodes affected by the constraint C. A zero mean gaussian noise is added to these activations a'_c. Subsequently, the samples and their respective noisy activations a'_c are fed as input to the critic network. The Bellman error is computed for the critic network from the output of the network and the new target computed from the output and the reward. This error is used to train the critic network. A separate gradient is computed as the derivative of the output with respect to the activation inputs. This gradient maximizes the reward function, i.e., the output of the critic network, with respect to the activations. Therefore the actor network has to learn to maximize the reward function by modifying the activations.

This gradient may be applied to the actor network in two ways. It can be used as the task error gradient in case of the classification problem with incomplete target data. In this case the output layer activations a_o are the affected nodes. The actor network has to learn the activation class outputs for which the reward is maximized. In this case the gradient from the critic network is the sole gradient applied to the actor network.

In contrast, to solve the problem of lateral inhibitory sparsity constraint, this gradient is applied to the hidden layer nodes. The gradient from the critic network is combined with the task error or reconstruction error, backpropagated from the output layer, as a regularization term in the hidden layer.

$$\frac{\partial E}{\partial w} + \theta\left(\frac{\partial Q(S_t, a'_{c_t})}{\partial a'_{c_t}}\right)$$

where $\partial E/\partial w$ is the gradient element from the task error, and θ controls the weight of the sparsity term from the critic network. In this case the actor network has to learn activations in the hidden layer which, in addition to the actual reconstruction error, also have to maximize the reward accrued from the imposed structural constraint.

V. Classification with Incomplete Target Data

In traditional multi-label classification problems the network tries to minimize the mean squared error E between the output y and the target τ. The gradient is defined as

$$E = (\tau - y)^2$$
$$\frac{\partial E}{\partial y} = \tau - y \tag{4}$$

However, this is not possible for problems where the target τ is not defined. For example, there are problems where, instead of the actual target, there exists a correct or wrong boolean indicating whether the sample was correctly classified or not. The function can be represented as

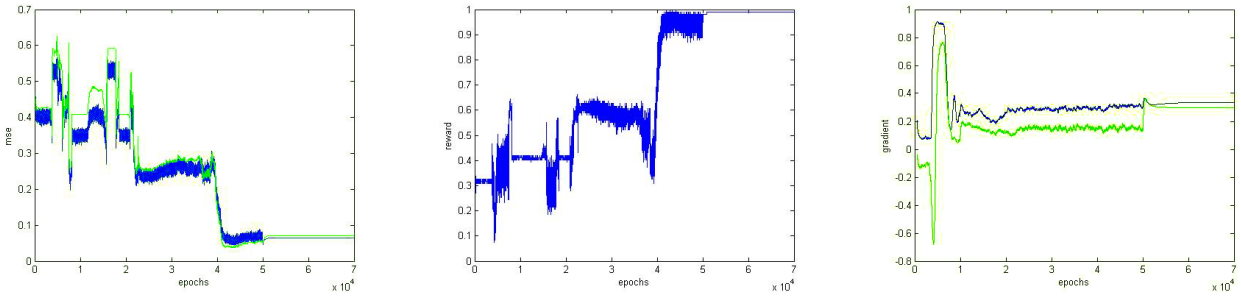
$$f(y) = \begin{cases} 1 & : C_y = C_\tau \\ 0 & : C_y \neq C_\tau \end{cases} \tag{5}$$

Here C_y is the class from the output y and C_t is the correct class of the input. In such cases the only information available is non-differentiable. The network is to be rewarded for correct classification and no reward for wrong classification. For such problems the system does not know what the actual class is nor does it know which output nodes correspond to which class. For the proposed model in this case the constraint to be satisfied is simply given by Equation 5 which provides the reward for the critic network. The nodes affected by the constraint are the output node activations, $a_c = a_o$, as shown in Figure 2. The critic network has to learn the constraint function such that the gradient in Equation 3 approximately equals

$$\frac{\partial Q(S_t, a'_{c_t})}{\partial a'_{c_t}} \approx \frac{\partial E}{\partial y}$$

A. Experiments

Experiments were done to compare the classification accuracy and mean-squared error achieved by classification using the proposed model and a traditional neural network. Training



(a) MSE (b) Reward (c) Gradient Comparison

Fig. 3: Proposed approach with incomplete target data. (a) Mean-squared error for the iris dataset, where the green line shows the validation error while the blue line shows training error. (b) Reward accrued over epochs. (c) Comparison of the approximated gradient vs the true gradient for a single sample and a single node over epochs, where the green line shows the approximated gradient from the critic network while the blue line shows the true gradient.

was done with a reward of 1 if correctly classified and zero if incorrectly classified and compared to the traditional classification network which had access to the actual target data. The classifier network was trained with 70% of the number of input nodes as hidden layer, and sigmoid activation functions for both networks. Since the reward term is either one or zero, sigmoid activation functions were also used in the critic network. A larger network might be used to increase performance of the task, for example increasing the number of layers and nodes would boost the performance of these tasks. However, a single hidden layer with few nodes was chosen in order to demonstrate the validity of the proposed approach.

The model was tested on three real world problems, by dividing the data into training, validation and testing segments.[1]

1) Iris dataset: The iris dataset has four input features and 150 samples in the dataset with three classes.

2) Thyroid dataset: The thyroid dataset has 21 input features and 7200 samples in the dataset with three classes.

3) White Wine Quality dataset: The white wine quality dataset has 11 input features and 3918 samples and 9 classes Table I shows the comparisons of accuracy and mean squared error achieved for the classification problems with the proposed model with incomplete data and traditional neural network classifiers with the entire dataset. The first two columns show the results from the proposed model while the traditional classifier is shown in the next two columns. This data shows

	Proposed		traditional	
Dataset	accuracy	MSE	accuracy	MSE
Iris	**96.67%**	0.0664	96.67%	0.0096
Thyroid	93.2%	0.037	**94.2%**	0.031
white Wine quality	**56.63%**	0.071	56.4%	0.067

TABLE I: Results from incomplete target data vs traditional network with complete target data

that despite having less information the proposed approach is capable of learning a similar quality solution. For the proposed model, Figure 3(a) shows the mean-squared classification error plot for the iris dataset. It takes some time for the critic network to model the reward function in order to provide an approximately correct gradient. There may be multiple solutions in the reward space of the critic network. Maximizing the reward function in the actor network might lead to going down towards a single solution and then flipping the class outputs if a new, better space is found via exploration. This might require redoing the gradient function. Thus, in the initial training stages, the actor network performs poorly. However, once the critic network models the reward, the mean-squared reconstruction error from the actor network is minimized.

Figure 3(b) shows the reward from the actor network structure over epochs. This shows the jumps made by the critic network whenever a space with a higher reward is identified via the added noise exploration.

Figure 3(c) compares the gradient from the critic network to the true gradient of the traditional classifier network.

$$\frac{\partial Q(S_t, a'_{c_t})}{\partial a'_{c_t}} \approx \frac{\partial E}{\partial y}$$

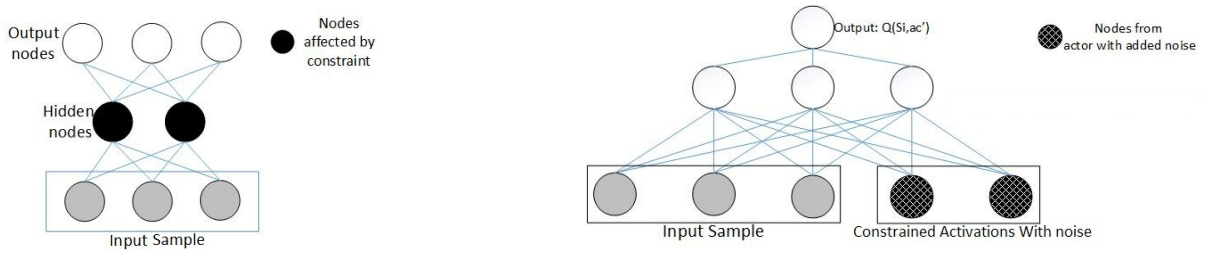
The gradient for a single sample and a single node over the training epochs is shown. The green line shows the gradient learned by the critic network while the blue line shows the true gradient computed with the entire target dataset. Figure 3(c) shows that the critic network initially provides a wrong gradient, however, after some time approximates the true gradient closely even with the incomplete target dataset.

VI. Proposed form of sparsity

Sparsity in an autoencoder forces the feature activations to be as close to zero as possible. This is usually achieved over the activation of a node over all samples[6]. The average activation of a hidden unit j over the training set of m samples is defined as

$$\hat{\rho_j} = \frac{1}{m} \sum_{i=1}^{m} [a_j^h(x^{(i)})]$$

[1]All datasets were accessed from UC Irvine Machine Learning Repository http://archive.ics.uci.edu/ml/



(a) Actor network

(b) Critic network

Fig. 4: Sparse autoencoder networks. (a) Actor network, shaded grey nodes indicate input nodes while constraint applied on hidden nodes. (b) Critic network, with input and noisy hidden node activations as input

A sparsity parameter ρ is defined as the target average activation of each node. In order to achieve sparsity, an additional penalty term in the form of KL-divergence is thus used.

$$KL(\rho||\hat{\rho_j}) = \rho log\frac{\rho}{\hat{\rho_j}} + (1-\rho)log\frac{1-\rho}{1-\hat{\rho_j}}$$

The gradient computed from this function is then used as a regularization parameter in addition to the gradient backpropagated for minimizing the reconstruction error.

$$\delta^h = \sum_{i=1}^{n}(W_i^h\delta^o)f'(a^h) + \beta\left(-\frac{\rho}{\hat{\rho_j}} + \frac{1-\rho}{1-\hat{\rho_j}}\right)$$

Here f' is the derivative of the activation function and W are the weights.

The goal of sparsity is to represent the data using as few values as possible. However traditional KL sparsity does not enforce any conditions on the maximum number of activations allowable to reconstruct a given sample. Although it achieves a similar form of sparsity indirectly, most features may fire for a few samples and remain close to zero for the rest, still achieving the desired average activation, but losing the essence of dimensionality reduction. Multiple features firing at the same time indicate that some discriminative property of the data is divided among all the features in order to keep the average activation of each node as low as possible, thus features learned overlap over the dataset. This results in inefficient and redundant features. The resulting feature set is sparse but lacks important structural insights of the data.

In order to learn non-overlapping discriminative and sparse features, it is often necessary to laterally inhibit features [17] [18]. However implementing such a constraint requires interconnections among nodes in the hidden layer. This results in additional computational complexity and complex network architecture which requires the update of a node to include the activations of other nodes in the layer. In such architectures the networks have minimal lateral connections, usually between immediate neighbors, in order to reduce the complexity but also preserve some lateral inhibition [2]. The proposed model, however, allows us to impose a novel structural constraint where the output of a node is influenced by every other node in the hidden layer. This allows for a much stricter constraint, which is not possible for traditional methods for lateral inhibitions. This constraint therefore enforces as few high activations as possible in the entire hidden layer to reconstruct the sample as opposed to lateral inhibitions among immediate neighbors. In contrast to traditional sparsity which strives to achieve close to zero average activation for a given node over all samples, this form of sparsity tries to minimize the number of features required to reconstruct a sample. This results in highly discriminative features and achieves the desirable property of locally expert features which fire while most other nodes remain close to zero.

The nodes affected by the constraint are the hidden layer nodes of the autoencoder, $a_c = a_h$, as shown in Figure 4. A node is considered to be active if the activation value is higher than a threshold parameter ϕ. The constraint is then encoded as a system of penalties, where the penalty for sample S_t and activations a'_c is given by

$$R(S_t, a_c) = \sum_{i=1}^{n}[a'_{c_i} > \phi]$$

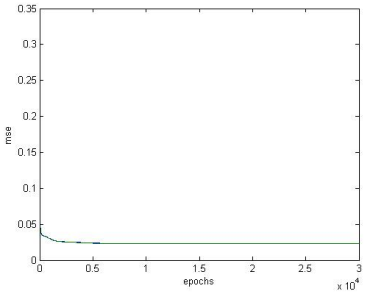
which is the number of nodes 'firing' for a given sample in the hidden layer with n nodes. In addition to the constraint being non-computable because of the lack of lateral connections among hidden nodes, the reward function is an L0 norm which is usually impossible to optimize directly. The critic network models a smooth approximation of the reward function which now has a gradient and can be optimized by the actor network. The gradient from the critic network output with respect to the hidden layer activations is used as a regularization parameter similar to traditional sparse autoencoders.

$$\delta^c = \sum_{i=1}^{n}(W_i^c\delta^o)f'(a^c) + \theta\left(\frac{\partial Q(x, a'_c)}{\partial a'_c}\right)$$

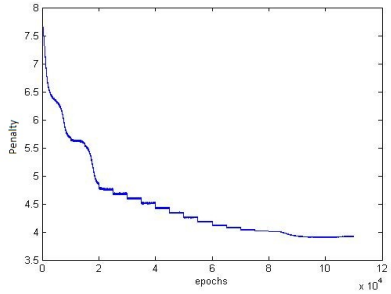
Where f' is the derivative of the activation function, W are the weights, and x are the input training samples.

A. Experiments

Experiments were done to compare the reconstruction accuracy and achievable sparsity of the proposed form of sparsity and traditional sparse methods using KL-divergence. The



(a) MSE



(b) Penalty

Fig. 5: Proposed form of sparsity. (a) Reconstruction mean-squared error for thyroid dataset (b) Penalty accrued over epochs

autoencoder network was trained with 70% of the number of input nodes as hidden layer. A larger network is needed to increase performance of the reconstruction task, however, the number of nodes for the hidden layer was chosen in order to demonstrate the validity of the proposed approach. The sigmoid function was used as activation functions of all nodes in the autoencoder. Since the penalty term is the number of nodes firing for a given sample, which can be more than one, softplus activation functions were used in the critic network for the hidden and output layers. The softplus activation function has the property of non-linear functions, whose output lies in the range of $(0, \infty)$.

Weight regularization is an important aspect of any sparse autoencoder system, since the activations can be learned to arbitrarily small values by simply learning higher weights. To force the system to learn proper sparse features, a weight reglarization penalty term was added to both models in order to keep the weights low. The model was tested on five real world problems, by dividing the data into training, validation and testing segments. Wight parameters $0.1 - 2$ as the range of θ and $0.2 - 16.8$ as β value ranges were used.

1) Abalone dataset: The abalone dataset is a highly imbalanced dataset with over 80% of the samples belonging to the same class. The dataset has 8 input features and 4177 samples. The autoencoder network was trained with six hidden nodes.

2) White Wine dataset: The wine white dataset has 11 input features and 3918 samples in the dataset. The autoencoder network was trained with eight hidden nodes.

3) Wine dataset: The wine dataset has 13 input features and 178 samples in the dataset. The autoencoder network was trained with 10 hidden nodes.

4) Glass dataset: The glass dataset has 9 input features and 214 samples in the dataset. The autoencoder network was trained with seven hidden nodes.

5) Thyroid dataset: The thyroid dataset has 21 input features and 7200 samples in the dataset. The autoencoder network was trained with 15 hidden nodes.

The results of reconstruction and the minimum achievable sparsity for the problems are shown in Table II. The proposed model is able to achieve better reconstruction accuracies for many cases compared to the traditional form of sparsity while achieving much more compact and sparser representations. For the proposed model, the first column represents the minimum sparsity achieved which is the percentage of fires over the entire test sample set for every node, while the second column shows the mean-squared reconstruction error achieved for the sparsity. Similarly, the third and fourth column represent the minimum sparsity achieved and mean-squared reconstruction error for the traditional form of sparsity.

	Proposed sparsity		Traditional sparsity	
Dataset	minimum sparsity achieved	MSE	minimum sparsity achieved	MSE
Abalone	0.1663	**0.0076**	0.4285	0.0119
Wine white	0.11	**0.0035**	0.4908	0.0037
wine	0.0972	**0.0089**	0.4514	0.0124
glass	0.00	0.0087	0.2716	**0.0086**
thyroid	0.01	0.0123	0.5495	**0.0118**

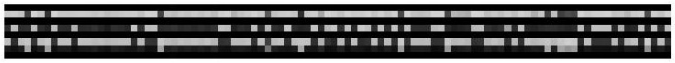
TABLE II: Comparison of the two forms of sparsity

A separate experiment was done using the thyroid dataset, which has 21 input dimensions, with eight hidden nodes. This forces the autoencoder networks to learn some features with higher activation values since the feature vector is not over complete. This was done to compare the type of features learned by the two approaches.

For the proposed model, Figure 5(a) shows the mean-squared reconstruction error plot for the thyroid dataset. Figure 5(b) shows the decrease in penalty of the actor network over epochs.

Figure 6 shows the features learned for 100 samples in the dataset. Figure 6(a), shows the features learned from the proposed approach while Figure 6(b) shows the features learned for the traditional sparse model. As shown in Figure 6(a) the features learned by the proposed model are much more discriminative as evident in features from node pairs (four, six) and two,seven where each pair never fires simultaneously.

In contrast, the traditional sparse system as shown in Figure 6(b) learns features which fire together, i.e., the discriminative property of the dataset is divided into multiple features in order to keep the average activation value of each feature as low



(a) Features learned by proposed lateral inhibitions approach

(b) Features learned by traditional sparsity constraint

Fig. 6: Thyroid dataset features. x-axis represents sample activations, y-axis represents individual nodes, intensity represents value of activations of a node for the given sample with white being almost one while black denoting non-firing nodes. (a) Proposed lateral inhibition sparsity constraint approach (b) Traditional KL-divergence sparsity constraint approach

as possible. Figure 6(a) shows that, the proposed approach is able to push down the activations of unneeded features, namely one,three,five and eight, towards zero in order to keep the number of activations per sample low.

VII. Conclusion

This paper presents an approach to solve problems without an inherent gradient by using an actor critic model. It is shown to be able to solve tasks which have an inherently non-differentiable or non-computable performance metric. The gradient is learned from a system of rewards from the imposed constraints function and is subsequently used to train the actual network. A novel form of lateral inhibition sparsity is also proposed which learns sparse features using as few activations as possible to reconstruct a given sample. It does this without interconnections among nodes in the hidden layer which made it unusable earlier with regards to computational complexity. Experiments evaluated on real world data show that this form of lateral sparsity is able to reconstruct the data better and with a much sparser representation.

VIII. Future Work

Given the applicability of the proposed approach, further exploration is necessary in order to assess the performance of the proposed form of sparsity in deep neural networks.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under Grant No. IIS-1551312.

References

[1] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[2] A. D. Szlam, K. Gregor, and Y. L. Cun, "Structured sparse coding via lateral inhibition," in *Advances in Neural Information Processing Systems*, 2011, pp. 1116–1124.

[3] N. Z. Shor, *Minimization methods for non-differentiable functions*. Springer Science & Business Media, 2012, vol. 3.

[4] J. Platt *et al.*, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[5] F. R. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 6.

[6] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, pp. 1–19, 2011.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[8] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," *Advances in neural information processing systems*, pp. 1038–1044, 1996.

[9] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPS*, vol. 99, 1999, pp. 1057–1063.

[10] G. Lever, "Deterministic policy gradient algorithms," 2014.

[11] S. Kakade, "A natural policy gradient." in *NIPS*, vol. 14, 2001, pp. 1531–1538.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] M. N. Katehakis and A. F. Veinott Jr, "The multi-armed bandit problem: decomposition and computation," *Mathematics of Operations Research*, vol. 12, no. 2, pp. 262–268, 1987.

[15] J. Langford and T. Zhang, "The epoch-greedy algorithm for multi-armed bandits with side information," in *Advances in neural information processing systems*, 2008, pp. 817–824.

[16] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine learning*, vol. 84, no. 1-2, pp. 137–169, 2011.

[17] J. Mutch and D. G. Lowe, "Object class recognition and localization using sparse features with limited receptive fields," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 45–57, 2008.

[18] W. E. Vinje and J. L. Gallant, "Sparse coding and decorrelation in primary visual cortex during natural vision," *Science*, vol. 287, no. 5456, pp. 1273–1276, 2000.