Self taught Learning

Based on the paper "Self-taught learning: Transfer Learning from Unlabeled Data"

Prateekshit Pandey

BTech CSE, 2nd year

January 3, 2014

Basics

- The basic study of learning starts from neurons (McCulloch-Pitts neuron model).
- A set of neurons when assigned to work on the same set of features, it forms a **layer of neurons for a neural network.**

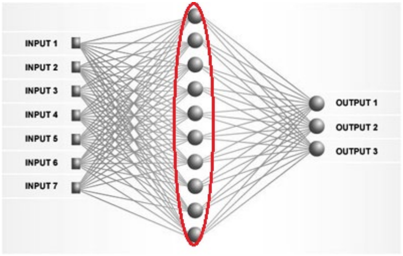

Figure 1 : A sample neural net

Feed forward and back-propagation

- Feed forward: Calculate activation layer of each layer and pass it on to the next layer, and thus, calculate the output layer.
- Back Propagation: Starting from the output layer, calculate the error for each layer.
  - $\delta^L = -(y - a^{(L)}). * f'(z^{(L)})$
  - $\delta^k = ((W^{(k)})^T.\delta^{k+1}). * f'(z^{(L)})$ for k = L-1, L-2, ... , 3, 2
- The cost function for a neural network is given by a one-half squared error error function.
  - $J(W, b) = \frac{1}{m} \sum_i^m (\frac{1}{2} || h_{W,b}(x^{(i)}) - y^{(i)} ||^2)$
- Gradient can be calculated using the errors calculated for each layer. This gradient can then be input to any optimization algorithm like gradient descent or L-BFGS.
  - $\nabla_{W^{(l)}} J(W, b) = \delta^{(l+1)} (a^{(l)})^T$
  - $\nabla_{b^{(l)}} J(W, b) = \delta^{(l+1)}$

Real world problems

- Perform speaker identification, provided unlimited access to natural sounds
- Perform classification of elephants and rhinos, provided unlimited access to natural images
- Perform email foldering of ICML reviewing emails and NIPS reviewing emails, provided unlimited access to news articles (text).
- Conclusion: always a mix of labeled and unlabeled data.

Problems faced

- **Labeled data**: difficult and expensive to obtain. Learning on a small data set may result in not being able to generalize over a larger data set.
- **Unlabled data**: expensive to find unlabeled data with desired class labels.
- **Motivation**: exploit the abundance of unlabeled data to generalize over a larger scale of data.

Previous algorithms and their shortcomings

- **Supervised learning**: works perfectly well if large amount of labeled data is provided, but fails to generalize well in case of scarcity of labeled data.
- **Semi-supervised learning**: needs labeled as well as unlabeled data for learning; assumes that the unlabeled data can be labeled with the same labels as the classification task.
- **Transfer learning**: typically requires transfer of knowledge from one supervised task to another, thus it requires additional labeled data.
- **Idea**: Transfer knowledge from unalabeled data.

Advantages and further motivations

- Use unlabeled data (from the same domain) without any restrictions.
- More accurately reflects how humans may learn, since much of human learning is believed to be from unlabeled data.

Problem formalisation

- Number of classes to classify data: C
- A set of m labeled examples: $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), ..., (x_l^{(m)}, y^{(m)})\}$ where $x_l^{(i)} \epsilon R^n$ and $y^{(i)} \epsilon \{1, 2, ..., C\}$
- A set of k unlabeled example: $\{x_u^{(1)}, x_u^{(2)}, ..., x_u^{(k)}\}$ where $x_u^{(i)} \epsilon R^n$
- The learning algorithm outputs a hypothesis $h : R^n \rightarrow \{1, 2, ..., C\}$
- The hypothesis function tries to mimic he input-output relationship represented by the labeled training data.
- This hypothesis function is tested under the same distribution from which labeled data was drawn.

Learning high level features - I

- We start with using the large unlabeled data to learn a higher level, more succint representation of the inputs.
- In case of images:
    - The inputs $x_u^{(i)}$ are vectors of pixel intensities of the images; the algorithm will try to learn 'basic elements' of the image.
    - The 'basic elements' can include some strong correlation between rows of pixels. Thus, it will be able to learn *edges*.
    - Thus, we will be able to present an image in terms of its edges, rather than raw pixel intensities.
- By applying learned representation to the labeled data, we obtain a higher level representation of the data. This makes the task of supervised learning much easier.

Learning high level features - II

- Following a modified version of sparse coding by Olshausen & Field (1996).
- Optimization objective:
  minimize$_{b,a} \sum_i^k ||x_u^{(i)} - \sum_j a_j^{(i)} b_j||_2^2 + \beta ||a^{(i)}||_1$
  - Number of bases: s
  - Basis: $b = \{b_1, b_2, ..., b_s\}; b_j \epsilon R^n$
  - Activations: $a = \{a^{(1)}, a^{(2)}, ..., a^{(k)}; a_j^{(i)} \epsilon R^s\}$
  - The number of bases s can be much larger than s.
- The optimization objective balances two terms:
  - The first quadratic term pushes each $x_u^{(i)}$ to be reconstructed well as a weighted linear combination of the bases.
  - It encourages the activation to have a low L_1 norm, thus encouraging the activations to be **sparse**.

Unsupervised Feature Construction

- It is often quite easy to obtain large amounts of unabeled data that shares several salient features with the classification task of interest.
- After learning a set of bases b from the unlabeled data, we compute the features $\hat{a}(x_l^{(i)})$ for the labeled data.
- We do this by solving the following optimization problem: $\hat{a}(x_l^{(i)}) = argmin_{a^{(i)}} ||x_l^{(i)} - \sum_j a_j^{(i)} b_j||_2^2 + \beta ||a^{(i)}||_1$
- Since, it is a L1 regularized optimization, we obtain a sparse representation of the labeled data

Algorithm: Self-taught learning via Sparse Coding

inputLabeled training set
50 mmT $= \{(x_l^{(1)}, y_l^{(1)}), (x_l^{(2)}, y_l^{(2)}), ..., (x_l^{(m)}, y_l^{(m)})\}$
Unlabeled data $\{x_u^{(1)}, x_u^{(2)}, ..., x_u^{(k)}\}$
output Learned classifier for the classification task.
algorithm Using unlabeled data $\{x_u^{(i)}\}$, solve the optimization problem to obtain bases b. Compute features for the classification task to obtain a new labeled training set $\hat{T} = \{(\hat{a}(x_l^{(i)}), y^{(i)})_{i=1}^m\}$, where
$\hat{a}(x_l^{(i)}) = argmin_{a^{(i)}} ||x_l^{(i)} - \sum_j a_j^{(i)} b_j||_2^2 + \beta ||a^{(i)}||_1$
Learn a classifier by applying a supervised learning algorithm (eg. SVM) to the labeled training set $\hat{T}$.
result the learned classifier C.

Comparison with other methods

- Every self-taught learning algorithm must be able to detect some structure using the unlabeled data.
- **Principal Component Analysis (PCA)**: identifies a lower dimensional subspace of maximal variation within the unlabeled data. The top $T \leq n$ principal components $b_1, b_2, ..., b_T$ are the solution to the optimization problem:
  $minimize_{b,a} \sum_i ||x_u^{(i)} - \sum_j a_j^{(i)} b_j||_2^2$
  such that $b_1, b_2, ..., b_T$ are orthogonal
- PCA seems convenient because it can be solved easily using standard numerical software. But, as compared to the sparse encoder, it has some limitations:
  - PCA results in a linear feature extraction; features $a_j^{(i)}$ are simply a linear function of the input. Sparse coding features $\hat{a}(x)$ are inherently non-linear.
  - PCA assumes bases to be orthogonal, hence the number of PCA features cannot exceed n. This is not a limitation in sparse coding.

Experiments: procedure followed

- For computational reasons, the unlaabeled data was preprocessed by applying PCA to reduce its dimensions.
- The sparse coding based algorithm was then applied in the resulting principal component space.
- The learned features were then used to construct features for each input from the supervised classification task.

Experiment data - I

Domain	Unlabeled data	Labeled data	Classes	Raw features
Image Classification	10 images of outdoor scenes	Caltech101 image classification dataset	101	Intensities in 14x14 pixel patch
Handwritten character recognition	Handwritten digits (0-9)	Handwritten english characters (a-z)	26	Intensities in 28x28 character/digit image

Table 1 : Details of self-taught learning applications evaluated in the experiments conducted by Raina et al (2009)

Experiment data - II

Domain	Unlabeled data	Labeled data	Classes	Raw features
Font character recognition	Handwritten English characters (a-z)	Font characters (a/A - z/Z)	26	Intensities in 28x28 character image
Song genre classification	Song snippets from 10 genres	Song snippets from 7 diff. genres	7	Long frequency spectogram over 50ms time window

Table 2 : Details of self-taught learning applications evaluated in the experiments conducted by Raina et al (2009)

Experiment data - III

Domain	Unlabeled data	Labeled data	Classes	Raw features
Webpage classsification	100,000 news articles (Reuters newswire)	Categorized webpages (from DMOZ hierarchy)	2	Bag-of-words with 500 words
UseNet article classification	100,000 news articles (Reuters newswire)	Categorized UseNet posts (from SRAA dataset)	2	Bag-of-words with 377 words

Table 3 : Details of self-taught learning applications evaluated in the experiments conducted by Raina et al (2009)

Results: Accuracy on the self-taught learning tasks

Domain	Training set size	Unlabeled SC	Labeled PCA	Labeled SC
Handwritten	100	39.7%	36.2%	31.4%
	500	58.5%	50.4%	50.8%
	5000	73.1%	73.5%	73.0%
Font char	100	7.0%	5.2%	5.1%
	500	16.6%	11.7%	14.7%
	1000	23.2%	19.0%	22.3%
Webpages	4	64.3%	55.9%	53.6%
	10	75.9%	57.0%	54.8%
	20	80.4%	62.9%	60.5%
UseNet	4	63.8%	60.5%	50.9%
	10	68.7%	67.9%	60.8%

Table 4 : Accuracy on the self-learning tasks observed in the experiments conducted by Raina et al (2009)

Autoencoders: introduction

An **autoencoder** neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs.

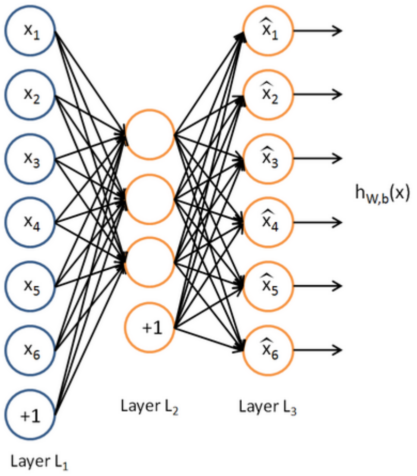

Figure 2 : Autoencoder

Autoencoders: introduction

- The autoencoder tries to learn a function $h_{W,b}(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x.]
- By placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data.
    - If the number of layers in the middle layer is less than the input, then it functions as a compressed representation of the input.
    - If the number of layers in the middle layer is more than the input, then it functions as a sparse representation of the input.

How it works

- By backpropagation, the error for the middle layer, or encoding layer, will be given by: $\delta_i^{(2)} = \left(\sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)}\right) f'(z_i^{(2)})$
- We introduce a term: $\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} \left[a_j^{(2)}(x^{(i)})\right]$ and enforce $\hat{\rho}_j = \rho$ where ρ is the sparsity parameter ans is kept near to zero (0.05). The sparsity parameter ensures that the activations are near to zero, and hence, sparse.
- Because of the sparsity parameter, a penalty term is added to the delta calculation for the encoding layer: $\delta_i^{(2)} = \left(\left(\sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)}\right) + \beta\left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i}\right)\right) f'(z_i^{(2)})$