Why Does Unsupervised Pre-training Help Deep Learning?

Peter Glöckner
Outline

- 1. Introduction
- 2. Hypothesis
- 3. Experimental Setup
- 4. Experiments
- 5. Online Learning Experiments
- 6. Conclusion
Introduction

- Standard training
  - Gradient-based optimization
    - n-dimensional function where n is the number of weights
    - Figure 1: 2 dimensions, convex function

Figure 1: [1]
Introduction

- Challenges of Deep Learning
  - model with many layers of adaptive parameters
  - highly non-convex objective function
  - potential for many distinct local minima
  - standard training schemes tend to place the parameter in regions of the parameter space that generalize poorly

Figure 2: [2]
Introduction

- Better results when splitting the training into two phases
  - 1) unsupervised pre-training:
    - Greedy layer-wise
    - Each layer learns a nonlinear transformation of its input, that captures the main variations in its input
  - 2) supervised fine-tuning:
    - The deep architecture is fine-tuned with respect to a supervised training criterion with gradient-based optimization

- Why does unsupervised pre-training help?
Hypothesis

- Preconditioning hypothesis

- Standard training
  - for a given layer weights are initialized using random samples from uniform $[-1/\sqrt{k},1/\sqrt{k}]$ where $k$ is the number of connections that a unit receives from a previous layer

- Unsupervised pre-training
  - acts as a kind of network pre-conditioner
  - putting the parameter values in the appropriate range for further supervised training

- We should get the same result when we
  - Select weight and biases according to the distribution obtained after supervised pre-training
Hypothesis

- optimization hypothesis
  - A gradient-based optimization should end in the apparent local minimum of whatever basin of attraction we start from
  - Unsupervised pre-training puts us in regions of the parameter space where basins of attractions run deeper
  - Better optimization
    → Achieving lower training costs
Hypothesis

- Regularization hypothesis

- Regularization effect is a consequence of the pre-training procedure
  - Establishing an initialization point of the fine-tuning procedure inside a region of the parameter space in which the parameters are henceforth restricted
  - Local basin of attraction of the supervised fine-tuning cost function
    (Defining a particular initialization point implicitly imposes constraints on the parameters)
  - Introducing bias towards configurations of the parameter space that are useful for unsupervised learning

- Observations we expect in the experiments
  - The two sets of models with and without unsupervised pre-training cover different regions in the parameter space
  - Better generalization/lower error on the test set
  - Not necessarily achieving lower training error (possibly worse)
  - Minimizing variance
Hypothesis

- pre-training restricts the parameters to particular regions
  - those that correspond to capturing structure in the input distribution $P(X)$

- unsupervised training criteria optimized during unsupervised pre-training
  - layers are trained to represent the dominant factors of variation in the data
  - form at each layer a representation of $X$ consisting of statistically reliable features of $X$
  - leveraging knowledge of $X$
  - if the pre-training strategy is effective, some of these learned features of $X$ are also predictive of $Y$
  - learning $P(X)$ is helpful for learning $P(Y|X)$
Hypothesis

- What we expect in online learning settings
  - the beneficial generalization effects due to pre-training do not appear to diminish as the number of labeled examples grows very large (contrary to classical regularizers)
  - non convexity of the objective function
  - dependency of the stochastic gradient descent method on example ordering
    - → starting point of a non-convex optimization matters
Experimental Setup

- **Datasets**
  - MNIST: handwritten digits in gray-scale
  - InfiniteMNIST: extension of MNIST
  - Shapeset: images of triangles, squares

- **Models (each with 1-5 hidden layers)**
  - standard feed-forward multi-layer neural networks
  - Deep Belief Networks (DBN)
  - Stacked Denoising Auto-Encoders (SDAE)

- **Hyperparameters**
  - Number of hidden units
  - ...

- **Random initialization**
  - for a given layer weights are initialized using random samples from uniform
    \([-1/sqrt(k),1/sqrt(k)]\) where k is the number of connections that a unit receives from a previous layer
Experiments

- Experiments without (left) and with (right) unsupervised pre-training
  - 400 random initialization seeds
  - MNIST dataset
  - Model failed to converge without supervised pre-training and 5 layers
  - Blue box: top and bottom quartiles
  - Red points: outliers

Figure 4: [4]
Experiments

- Unsupervised pre-training
  - Lower test classification error
  - Robustness to random initialization (variance stays same, bad outliers growing slowly)
  - Reduced variance supports regularization hypothesis (not explainable with a pure optimization effect)
- Without unsupervised pre-training
  - Increasing test error and variance when adding more layers
  - Increasing depth → Increasing probability of finding poor apparent local minima

![Figure 5: [4]](image)
Experiments

- Visualization shows to which input the units from the different layers most respond to
  - Model: DBN
  - Dataset: InfiniteMNIST

- Figure 6
  - Left to right: units from 1st, 2nd and 3rd layer
  - Top: after pre-training
  - Bottom: after pre- and supervised training

- Figure 7
  - After training without pre-training

Figure 6: [4]
Figure 7: [4]
Experiments

- After pre-training supervised fine-tuning does not change the weights significant way
  - Stuck in a certain region of weight space
  - Supervised training has more effect on the deeper layers
  - Features increase in complexity as we add more layers
- Consistent with the predictions made by our hypothesis
  - unsupervised pre-training can “lock” the training in a region of the parameter space that is essentially inaccessible for models that are trained in a purely supervised way
- Displaying only one image for each feature does not show the set of patterns on which the feature is highly active (or highly inactive)
- not useful to show how these strategies are influenced by random initialization
Experiments

- Visualization of Model Trajectories During Learning
  - Compare function represented by each network
  - Approximated function with a finite number of inputs
  - Concatenate all its outputs as one vector
  - One vector for each model at each training iteration
  - Map these vectors with a dimensionality reduction algorithm to a two-dimensional space
- Color from dark blue to cyan indicates a progression in training iterations
- 50 networks with and 50 without pre-training as supervised training proceeds over MNIST

Figure 8: [4]
Experiments

- Pre-trained and not pre-trained models start and stay in different regions of function space
- Trajectories diverge at the end of training
  - Different initialization seed move into a different local minimum
- Figure 9: different dimensionality reduction algorithm (focusing on global structure)

Figure 9: [4]
Experiments

- Analyze models obtained at the end of training
  - Dataset: Shapeset
  - Stepping in parameter space in 7 random gradient directions
  - Top/Bottom: Without/With pre-training
  - Left to Right: 1, 2, 3 hidden layers
- error landscape is a bit flatter in the case of unsupervised pre-training/deeper architectures
- Models seem to be near a local minimum

Figure 10: [4]
Experiments

- Pre-conditioning hypothesis
  - The range and marginal distribution from which we draw initial weights is responsible for the better results
- So we should get the same results when we
  - Perform unsupervised pre-training
  - Compute histograms for each layer's pre-trained weights and biases
  - Select weights and biases at random according to the histograms

<table>
<thead>
<tr>
<th>initialization</th>
<th>Uniform</th>
<th>Histogram</th>
<th>Unsup. Pre-tr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 layers</td>
<td>1.81 ± 0.07</td>
<td>1.94 ± 0.09</td>
<td>1.41 ± 0.07</td>
</tr>
<tr>
<td>2 layers</td>
<td>1.77 ± 0.10</td>
<td>1.69 ± 0.11</td>
<td>1.37 ± 0.09</td>
</tr>
</tbody>
</table>

Table 1: [4]
Experiments

- Test and training error for 400 models without (blue) and with (red) pre-training on MNIST
  - Since training error tend to decrease, the trajectories run from right to left
  - Only for 1 hidden layer unsupervised pre-training reaches lower training cost (better optimization)
  - at the same training costs, the pre-trained models have lower test costs (better generalization)

Figure 11: [4]
Experiments

- Experiment supports regularization hypothesis
- unsupervised pre-training
  - decreases variance
  - introduces a bias (towards “better” parameter configurations)
- It seems restricting the possible starting points in parameter space to those that minimize the unsupervised pre-training criterion
  - restricts the set of possible final configurations for parameter values
Experiments

- Relationship between units per layer and test error
- Regularization hypothesis suggest decreasing effectiveness with decreasing layer size
  - small networks have a limited capacity so further restricting it can harm generalization (extra regularization effect)
    (Less probable that useful input transformation created by unsupervised pre-training are included)

- Experiment
  - Unsupervised pre-training seems to help deeper/ hurt smaller networks
  - generalization error increases with decreasing number of units (increases more with unsupervised pre-training)
Experiments

- Support for the optimization hypothesis in the literature
  - the reported training and test errors were lower for pre-trained networks
  - test with early stopping
  - early stopping is itself a regularizer and it can influence greatly the training error that is obtained

- Recreate experiment without early stopping
  - Higher training error/ Lower generalization error for pre-trained networks
  - maybe early stopping prevented the networks without pre-training from moving too much towards their apparent local minimum

Figure 13: [4]
Online Learning Experiments

- Effect of pre-training with very large datasets (InfiniteMNIST)
- Online classification error (computed as an average over a block of last 100,000 errors)
  - Predict class for training example first and use prediction for calculating error
  - Use example for training
- Advantage of pre-training does not vanish
- 3 layer no pre-training generalizes worse than 1 layer no pre-training
- 1-layer networks without pre-training should in principle be able to represent the input distribution as capacity and data grow
  - Number of hidden units chosen individually (so that the error is minimized)
  - Without pre-training, networks are not able to take advantage of the additional capacity

Figure 14: [4]
Online Learning Experiments

- Train 1-layer network with and without pre-training with 10 Million examples
- Use the same examples for calculating classification error (in the same order they were used for training)
- Both models are better at classifying more recently seen examples
- Unsupervised pre-training shows an optimization effect
- empirical distribution (defined by the training set) converges to the true data distribution
  - Better optimization strategies should have significant impact on the generalization

Figure 15: [4]
Online Learning Experiments

- Effect of example ordering
  - Dataset with 10 million examples
  - Train 10 models with the same dataset, but change the order of the first million examples (keep the others fixed)
  - Measure variance of the output
  - Repeat process but now vary the next million examples
  - ...

- Figure 16
  - Lower variance with supervised pre-training
  - Increasing variance at the end: last examples have a greater influence
  - $x = 0.25$: start of supervised training for pre-trained networks

Figure 16: [4]
Online Learning Experiments

- Only pre-train the bottom $k$ layers
  - Left: training vs test error on MNIST at each epoch of training
    - Pre-train more layers $\rightarrow$ better generalization, worse training error
  - RIGHT: online classification error on InfiniteMNIST

![Graphs showing training vs test error and online classification error](image)

Figure 17: [4]
Conclusion

- Some effects of Unsupervised pre-training are similar to the effects of a good regularizer
- That the effect of unsupervised pre-training does not go away with more training data is not an effect of classical regularizers
- Network with pre-training has lower training error on a very large data set
  → results are still consistent with our hypothesis

Next steps
- More experiments for a better understanding of unsupervised pre-training
- Use different datasets, architectures, algorithms, ...
- Questions ...
References