Greedy Layer-Wise Training of Deep Networks

Yoshua Bengio, Pascal Lamblin, Dan Popovici,
Hugo Larochelle
U. Montreal

December 5th 2006

Thanks to: Yann Le Cun, Geoffrey Hinton,
Olivier Delalleau, Nicolas Le Roux
Motivation: AI ⇒ learning high-level abstractions
⇒ highly-varying functions
⇒ non-local + deep architecture

- Principle of greedy layer-wise unsupervised initialization
- Deep Belief Networks
- Deep Multi-layer Neural Networks
- Experimental study: why this principle works
- Extensions of Deep Belief Networks
Ambitious goal: using ML to reach AI

**AI tasks**: visual and auditory perception, language understanding, intelligent control, long-term prediction, understanding of high-level abstractions...

Remains elusive! (did we turn our back on it?)

3 considerations:
- computational efficiency
- statistical efficiency
- human-labor efficiency efficiency

Here: focus on algorithms associated with broad priors (i.e., non-parametric) with the hope of discovering principles applicable to vast array of tasks within AI, with no need of hand-crafted solutions for each particular task.
Grand Goal: AI

- Ambitious goal: using ML to reach AI
- AI tasks: visual and auditory perception, language understanding, intelligent control, long-term prediction, understanding of high-level abstractions...
- Remains elusive! (did we turn our back on it?)
- 3 considerations:
  - computational efficiency
  - statistical efficiency
  - human-labor efficiency efficiency
- Here: focus on algorithms associated with broad priors (i.e., non-parametric) with the hope of discovering principles applicable to vast array of tasks within AI, with no need of hand-crafted solutions for each particular task.
Ambitious goal: using ML to reach AI

*AI tasks*: visual and auditory perception, language understanding, intelligent control, long-term prediction, understanding of high-level abstractions...

Remains elusive! (did we turn our back on it?)

3 considerations:
- computational efficiency
- statistical efficiency
- student-labor efficiency

Here: focus on algorithms associated with broad priors (i.e., non-parametric) with the hope of discovering principles applicable to vast array of tasks within AI, with no need of hand-crafted solutions for each particular task.
Depth of Architectures

Depth = number of **levels** of composition of adaptable elements:

- kernel machines: shallow
- boosting: generally shallow
- multi-layer neural networks: usually shallow, can be deep?
- decision trees: deep but local estimators (curse of dim.)
- parametric graphical models: human-labor intensive

Non-parametric ones can theoretically approximate any continuous function. But **how efficiently?** (computational, statistical)
Inefficiency of Shallow Architectures

Mathematical results from complexity theory of boolean circuits:

Functions representable compactly by a deep circuit often need circuits of exponential size to be represented by a shallow circuit (Hastad 1987)

Very fat shallow circuit
⇒ many adjustable elements
⇒ many examples needed

Number of levels should not be fixed but data-dependent.

*Brain has a deep architecture*
(Bengio et al 2006):

*Local* kernel machines (= *pattern matchers*) and decision trees partition the space and may need exponential nb of units, i.e. of examples

inefficient at representing *highly-varying functions*, which may otherwise have a compact representation.
What deep architectures are known? various kinds of multi-layer neural networks with many layers.

Except for a very special kind of architectures for machine vision (convolutional networks), deep architectures have been neglected in machine learning.

Why? Training gets stuck in mediocre solutions (Tesauro 92).

Credit assignment problem?

No hope?
Learning AI ⇒ learning abstractions

General principle: Greedily learning simple things first, higher-level abstractions on top of lower-level ones.

Implicit prior: restrict to functions that
1. can be represented as a composition of simpler ones such that
2. the simpler ones can be learned first (i.e., are also good models of the data).

Coherent with psychological literature (Piaget 1952).
We learn baby math before arithmetic before algebra before differential equations ... 

Also some evidence from neurobiology: (Guillery 2005) “Is postnatal neocortical maturation hierarchical?”.
Learning AI ⇒ learning abstractions

General principle: Greedily learning simple things first, higher-level abstractions on top of lower-level ones.

Implicit prior: restrict to functions that
1. can be represented as a composition of simpler ones such that
2. the simpler ones can be learned first (i.e., are also good models of the data).

Coherent with psychological literature (Piaget 1952).
We learn baby math before arithmetic before algebra before differential equations . . .

Also some evidence from neurobiology: (Guillery 2005) “Is postnatal neocortical maturation hierarchical?”.
Learning AI ⇒ learning abstractions

General principle: Greedily learning simple things first, higher-level abstractions on top of lower-level ones.

Implicit prior: restrict to functions that
1. can be represented as a composition of simpler ones such that
2. the simpler ones can be learned first (i.e., are also good models of the data).

Coherent with psychological literature (Piaget 1952).
We learn baby math before arithmetic before algebra before differential equations . . .

Also some evidence from neurobiology: (Guillery 2005) “Is postnatal neocortical maturation hierarchical?”.
Deep Belief Networks

Hinton et al (2006) recently introduced a deep graphical model that provides more evidence that this principle works:

- beats state-of-the-art statistical learning in experiments on a large machine learning benchmark task (knowledge-free MNIST)

See also Ranzato et al spotlight/poster tomorrow

Each layer tries to model distribution of its input (unsupervised training as Restricted Boltzmann Machine)

- $H =$ hidden causes,
  
  $P(h|x) =$ representation of $x$.

- Unsupervised greedy layer-wise initialization replaces traditional neural net random initialization.
Deep Belief Networks

Hinton et al (2006) recently introduced a deep graphical model that provides more evidence that this principle works:

- beats state-of-the-art statistical learning in experiments on a large machine learning benchmark task (knowledge-free MNIST)
  See also Ranzato et al spotlight/poster tomorrow

- Each layer tries to model distribution of its input (unsupervised training as Restricted Boltzmann Machine)

  \[ H = \text{hidden causes,} \quad P(h|x) = \text{representation of } x. \]

- Unsupervised greedy layer-wise initialization replaces traditional neural net random initialization.
Hinton et al (2006) recently introduced a deep graphical model that provides more evidence that this principle works:

- beats state-of-the-art statistical learning in experiments on a large machine learning benchmark task (knowledge-free MNIST)
- Each layer tries to model distribution of its input (unsupervised training as Restricted Boltzmann Machine)
- $H =$ hidden causes, $P(h|x) =$ representation of $x$.
- Unsupervised greedy layer-wise initialization replaces traditional neural net random initialization.

See also Ranzato et al spotlight/poster tomorrow
Hinton et al (2006) recently introduced a deep graphical model that provides more evidence that this principle works:

- beats state-of-the-art statistical learning in experiments on a large machine learning benchmark task (knowledge-free MNIST)
- See also Ranzato et al spotlight/poster tomorrow

- Each layer tries to model distribution of its input (unsupervised training as Restricted Boltzmann Machine)
- $H = \text{hidden causes, } P(h|x) = \text{representation of } x.$
- Unsupervised greedy layer-wise initialization replaces traditional neural net random initialization.
The principle of greedy layer-wise initialization proposed by Hinton can be generalized to other algorithms.

Initialize each layer of a deep multi-layer feedforward neural net as an autoassociator for the output of the previous layer.

Find $W$ which minimizes cross-entropy loss in predicting $x$ from $\hat{x} = \text{sigm}(W' \text{sigm}(Wx))$.

Feed its hidden activations as input to next layer.

Sigmoid and small weights (weight decay or stochastic gradient) prevent autoassociator from learning the identity.
The principle of greedy layer-wise initialization proposed by Hinton can be generalized to other algorithms.

Initialize each layer of a **deep multi-layer feedforward neural net** as an **autoassociator** for the output of previous layer.

Find $W$ which minimizes cross-entropy loss in predicting $x$ from $\hat{x} = \text{sigm}(W'\text{sigm}(Wx))$.

Feed its hidden activations as input to next layer.

Sigmoid and small weights (weight decay or stochastic gradient) prevent autoassociator from learning the identity.
The principle of greedy layer-wise initialization proposed by Hinton can be generalized to other algorithms.

Initialize each layer of a deep multi-layer feedforward neural net as an autoassociator for the output of previous layer.

Find $W$ which minimizes cross-entropy loss in predicting $x$ from
\[ \hat{x} = \text{sigm}(W' \text{sigm}(Wx)). \]

Feed its hidden activations as input to next layer.

Sigmoid and small weights (weight decay or stochastic gradient) prevent autoassociator from learning the identity.
The principle of greedy layer-wise initialization proposed by Hinton can be generalized to other algorithms.

Initialize each layer of a deep multi-layer feedforward neural net as an autoassociator for the output of previous layer.

Find $W$ which minimizes cross-entropy loss in predicting $x$ from $\hat{x} = \text{sigm}(W' \text{sigm}(Wx))$.

Feed its hidden activations as input to next layer.

Sigmoid and small weights (weight decay or stochastic gradient) prevent autoassociator from learning the identity.
The principle of greedy layer-wise initialization proposed by Hinton can be generalized to other algorithms.

Initialize each layer of a deep multi-layer feedforward neural net as an autoassociator for the output of previous layer.

Find $W$ which minimizes cross-entropy loss in predicting $x$ from $\hat{x} = \text{sigm}(W'\text{sigm}(Wx))$.

Feed its hidden activations as input to next layer.

Sigmoid and small weights (weight decay or stochastic gradient) prevent autoassociator from learning the identity.
Why not use supervised learning at each stage?

- Each layer is trained as the hidden layer of a supervised 2-layer neural net.
- After training the 2-layer neural net, discard output layer;
- Propagate data through new hidden layer, train another layer, etc.
Why not use supervised learning at each stage?

- Each layer is trained as the hidden layer of a supervised 2-layer neural net.
- After training the 2-layer neural net, discard output layer;
- Propagate data through new hidden layer, train another layer, etc.
Why not use supervised learning at each stage?

- Each layer is trained as the hidden layer of a supervised 2-layer neural net.
- After training the 2-layer neural net, discard output layer;
- Propagate data through new hidden layer, train another layer, etc.
Why not use supervised learning at each stage?

- Each layer is trained as the hidden layer of a supervised 2-layer neural net.
- After training the 2-layer neural net, discard output layer;
- Propagate data through new hidden layer, train another layer, etc.
### Experiments on Greedy Layer-Wise Wise Initialization

<table>
<thead>
<tr>
<th>Model</th>
<th>Train.</th>
<th>Test.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Belief Net, unsupervised pre-training</td>
<td>0%</td>
<td>1.2%</td>
</tr>
<tr>
<td>Deep net, autoassociator pre-training</td>
<td>0%</td>
<td>1.4%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>2.0%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>.004%</td>
<td>2.4%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>.004%</td>
<td>1.9%</td>
</tr>
</tbody>
</table>

**Classification error on MNIST digits benchmark** training, validation, and test sets, with the best hyper-parameters according to validation error.

Deep nets with 3 to 5 hidden layers.
Selects around 500 hidden units per layer.

Supervised greedy is **too greedy**.
Greedy unsupervised initialization works great.

Why 0 train error even with deep net / no-pretraining?
Experiments on Greedy Layer-Wise Initialization

<table>
<thead>
<tr>
<th>Model</th>
<th>Train Error</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Belief Net, unsupervised pre-training</td>
<td>0%</td>
<td>1.2%</td>
</tr>
<tr>
<td>Deep net, autoassociator pre-training</td>
<td>0%</td>
<td>1.4%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>2.0%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>0.004%</td>
<td>2.4%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>0.004%</td>
<td>1.9%</td>
</tr>
</tbody>
</table>

Classification error on MNIST digits benchmark training, validation, and test sets, with the best hyper-parameters according to validation error.

Deep nets with 3 to 5 hidden layers.
Selects around 500 hidden units per layer.

Supervised greedy is too greedy.
Greedy unsupervised initialization works great.

Why 0 train error even with deep net / no-pretraining?
<table>
<thead>
<tr>
<th>Model</th>
<th>Train Error</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Belief Net, unsupervised pre-training</td>
<td>0%</td>
<td>1.2%</td>
</tr>
<tr>
<td>Deep net, autoassociator pre-training</td>
<td>0%</td>
<td>1.4%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>2.0%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>0.004%</td>
<td>2.4%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>0.004%</td>
<td>1.9%</td>
</tr>
</tbody>
</table>

**Classification error on MNIST digits benchmark** training, validation, and test sets, with the best hyper-parameters according to validation error.

Deep nets with 3 to 5 hidden layers.
Selects around 500 hidden units per layer.

**Supervised greedy is too greedy.**
Greedy unsupervised initialization works great.

Why 0 train error even with deep net / no-pretraining?
Classification error on MNIST with 20 hidden units on top layer:

<table>
<thead>
<tr>
<th>Model</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Belief Net, unsupervised pre-training</td>
<td>.008%</td>
<td>1.5%</td>
</tr>
<tr>
<td>Deep net, autoassociator pre-training</td>
<td>0%</td>
<td>1.6%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>.59%</td>
<td>2.2%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>3.6%</td>
<td>5.0%</td>
</tr>
</tbody>
</table>

Because

1. last fat hidden layer did all the work
2. using a poor representation (output of all previous layers)

Yes it is really an optimization problem and a representation problem
Is it Really an Optimization Problem?

Classification error on MNIST with 20 hidden units on top layer:

<table>
<thead>
<tr>
<th>Method</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Belief Net, unsupervised pre-training</td>
<td>.008%</td>
<td>1.5%</td>
</tr>
<tr>
<td>Deep net, autoassociator pre-training</td>
<td>0%</td>
<td>1.6%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>.59%</td>
<td>2.2%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>3.6%</td>
<td>5.0%</td>
</tr>
</tbody>
</table>

Because

1. last fat hidden layer did all the work
2. using a poor representation (output of all previous layers)

Yes it is really an **optimization** problem
and a **representation** problem
Restricted Boltzmann Machines

Bi-partite Boltzmann machine:

\[ P(X = x, H = h) \propto e^{-\mathcal{E}(x, h)} = e^{x'b + h'c + h'Wx} \]

- Conditionals \( P(x|h) \) and \( P(h|x) \) easy to derive, and factorize.
- Contrastive divergence provides good estimator of log-likelihood gradient.
- Originally for binary variables; we extend it \textbf{easily} to continuous variables by slightly changing energy function and range of values (see poster).
Combining Supervised & Unsupervised

MNIST: nice clusters in the distribution
⇒ input distribution structure reveals the target class.
\[ f_1(x) = P(Y|x) \] related to \[ f_2(x) = P(x) \]

Otherwise? Simple solution:
combine supervised & unsupervised layer-wise greedy initialization.

Just add the two stochastic gradient updates.
Combining Supervised & Unsupervised

MNIST: nice clusters in the distribution
⇒ **input distribution structure reveals the target class.**

\[ f_1(x) = P(Y|x) \] related to \[ f_2(x) = P(x) \]

Otherwise? Simple solution:
**combine supervised & unsupervised layer-wise greedy initialization.**

Just add the two stochastic gradient updates.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle

NIPS*2006
Combining Supervised & Unsupervised

MNIST: nice clusters in the distribution

$\Rightarrow$ **input distribution structure reveals the target class.**

$f_1(x) = P(Y|x)$ related to $f_2(x) = P(x)$

Otherwise? Simple solution:

combine supervised & unsupervised layer-wise greedy initialization.

Just add the two stochastic gradient updates.
Combining Supervised & Unsupervised

MNIST: nice clusters in the distribution
⇒ input distribution structure reveals the target class.
\[ f_1(x) = P(Y|x) \text{ related to } f_2(x) = P(x) \]

Otherwise? Simple solution:
combine supervised & unsupervised layer-wise greedy initialization.

Just add the two stochastic gradient updates.
Combining Supervised & Unsupervised

MNIST: nice clusters in the distribution
⇒ **input distribution structure reveals the target class.**

$f_1(x) = P(Y|x)$ related to $f_2(x) = P(x)$

Otherwise? Simple solution:
combine supervised & unsupervised layer-wise greedy initialization.

Just add the two stochastic gradient updates.
More experimental results

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE</th>
<th>Cotton class. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN, Gauss inputs, partially sup</td>
<td>4.18</td>
<td>31.4%</td>
</tr>
<tr>
<td>DBN, Gauss inputs, unsup</td>
<td>4.19</td>
<td>35.8%</td>
</tr>
<tr>
<td>DBN, Bin inputs, partially sup</td>
<td>4.28</td>
<td>43.7%</td>
</tr>
<tr>
<td>DBN, Bin inputs, unsup</td>
<td>4.47</td>
<td>45.0%</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>.</td>
<td>45.0%</td>
</tr>
<tr>
<td>Deep Network, no pre-training</td>
<td>4.2</td>
<td>43.0%</td>
</tr>
</tbody>
</table>

MSE on Abalone task and classification error on Cotton task, showing improvement with Gaussian vs binomial units and partial supervision.
For AI ⇒ must learn **high level abstractions** efficiently
⇒ **deep architectures** (statistical efficiency)

- Deep architectures not trainable? computational efficiency?
  new methods appear to **break through the obstacle**
- Basic principle: **greedy layer-wise unsupervised** (or adding
  unsupervised and supervised criteria)
- **Principle works about as well with symmetric autoassociators** in feedforward neural net
- The **unsupervised part** is important: regularizes and makes
  sure to propagate most information about input, **purely supervised is too greedy**.
- Easy extensions of Deep Belief Nets: continuous-valued units
  / partially supervised initialization when input density is not
  revealing of target
- Come see the poster!
Conclusions

- For AI ⇒ must learn **high level abstractions** efficiently ⇒ **deep architectures** (statistical efficiency)
- Deep architectures not trainable? computational efficiency? new methods appear to **break through the obstacle**
- Basic principle: greedy layer-wise unsupervised (or adding unsupervised and supervised criteria)
- **Principle works about as well with symmetric autoassociators** in feedforward neural net
- The unsupervised part is important: regularizes and makes sure to propagate most information about input, **purely supervised is too greedy**.
- Easy extensions of Deep Belief Nets: continuous-valued units / partially supervised initialization when input density is not revealing of target
- Come see the poster!

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle  NIPS*2006
Conclusions

- For AI ⇒ must learn **high level abstractions** efficiently
  ⇒ **deep architectures** (statistical efficiency)

- Deep architectures not trainable? computational efficiency?
  new methods appear to **break through the obstacle**

- Basic principle: **greedy layer-wise unsupervised** (or adding unsupervised and supervised criteria)

- Principle works about as well with symmetric autoassociators in feedforward neural net

- The **unsupervised part** is important: regularizes and makes sure to propagate most information about input, **purely supervised is too greedy**.

- Easy extensions of Deep Belief Nets: continuous-valued units / partially supervised initialization when input density is not revealing of target

- Come see the poster!

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle  NIPS*2006
Conclusions

- For AI ⇒ must learn **high level abstractions** efficiently
  ⇒ **deep architectures** (statistical efficiency)
- Deep architectures not trainable? computational efficiency?
  new methods appear to **break through the obstacle**
- Basic principle: **greedy layer-wise unsupervised** (or adding unsupervised and supervised criteria)
- **Principle works about as well with symmetric autoassociators** in feedforward neural net
  - The **unsupervised part** is important: regularizes and makes sure to propagate most information about input, **purely supervised is too greedy**.
  - Easy extensions of Deep Belief Nets: continuous-valued units / partially supervised initialization when input density is not revealing of target
- Come see the poster!

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle  NIPS*2006
For AI ⇒ must learn **high level abstractions** efficiently
⇒ **deep architectures** (statistical efficiency)

Deep architectures not trainable? computational efficiency?
new methods appear to **break through the obstacle**

Basic principle: **greedy layer-wise unsupervised** (or adding unsupervised and supervised criteria)

**Principle works about as well with symmetric autoassociators** in feedforward neural net

The **unsupervised part** is important: regularizes and makes sure to propagate most information about input, **purely supervised is too greedy**.

Easy extensions of Deep Belief Nets: continuous-valued units / partially supervised initialization when input density is not revealing of target

Come see the poster!
For AI ⇒ must learn **high level abstractions** efficiently
⇒ **deep architectures** (statistical efficiency)

Deep architectures not trainable? computational efficiency? new methods appear to **break through the obstacle**

Basic principle: **greedy layer-wise unsupervised** (or adding unsupervised and supervised criteria)

**Principle works about as well with symmetric autoassociators** in feedforward neural net

The **unsupervised part** is important: regularizes and makes sure to propagate most information about input, **purely supervised is too greedy**.

Easy extensions of Deep Belief Nets: continuous-valued units / partially supervised initialization when input density is not revealing of target

Come see the poster!