Deep Learning の基礎と言語処理への応用

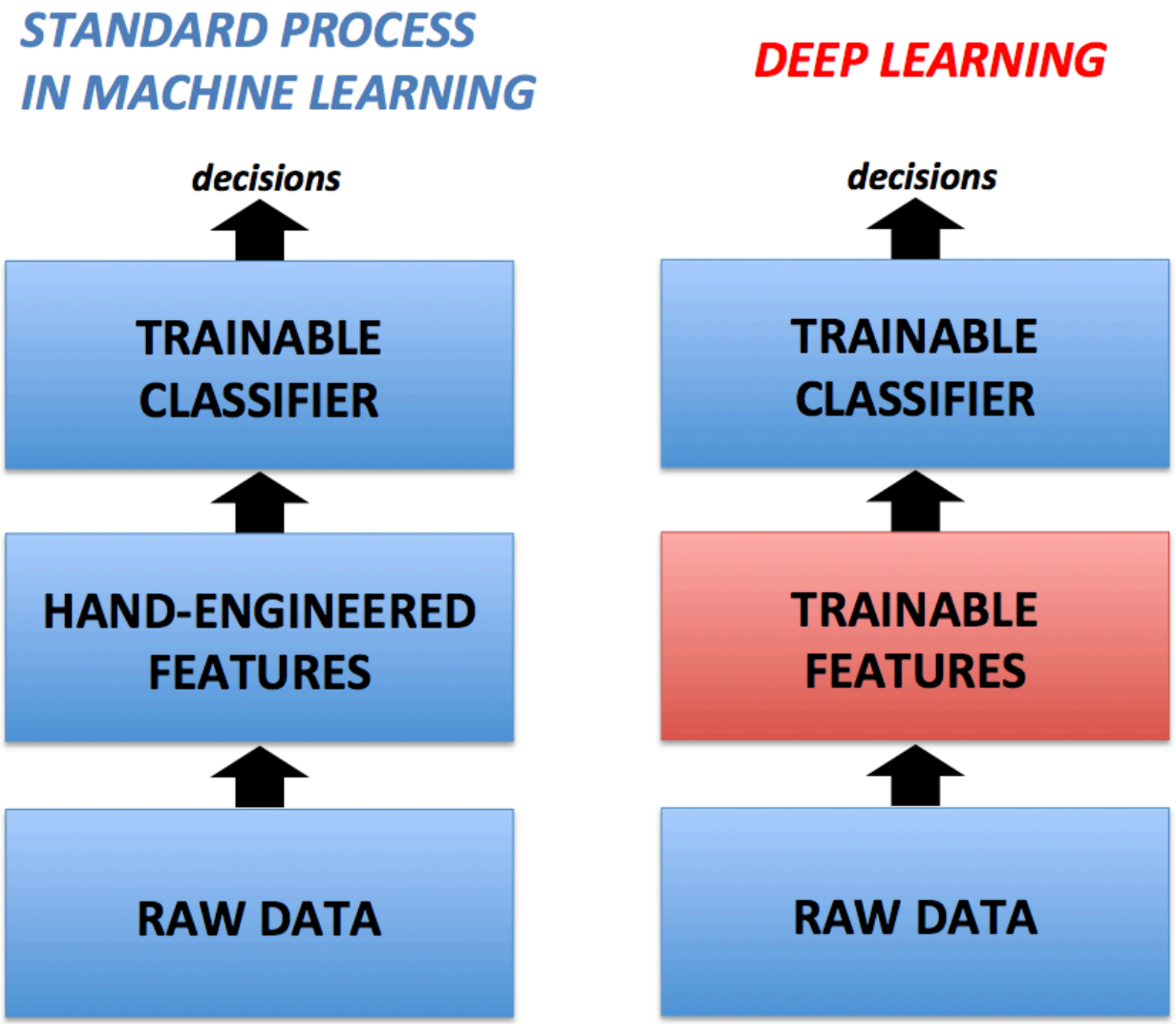
Kevin Duh

奈良先端科学技術大学院大学情報科学研究科

言語処理学会年次大会チュートリアル 2014/03/17

Deep Learning（深層学習）とは

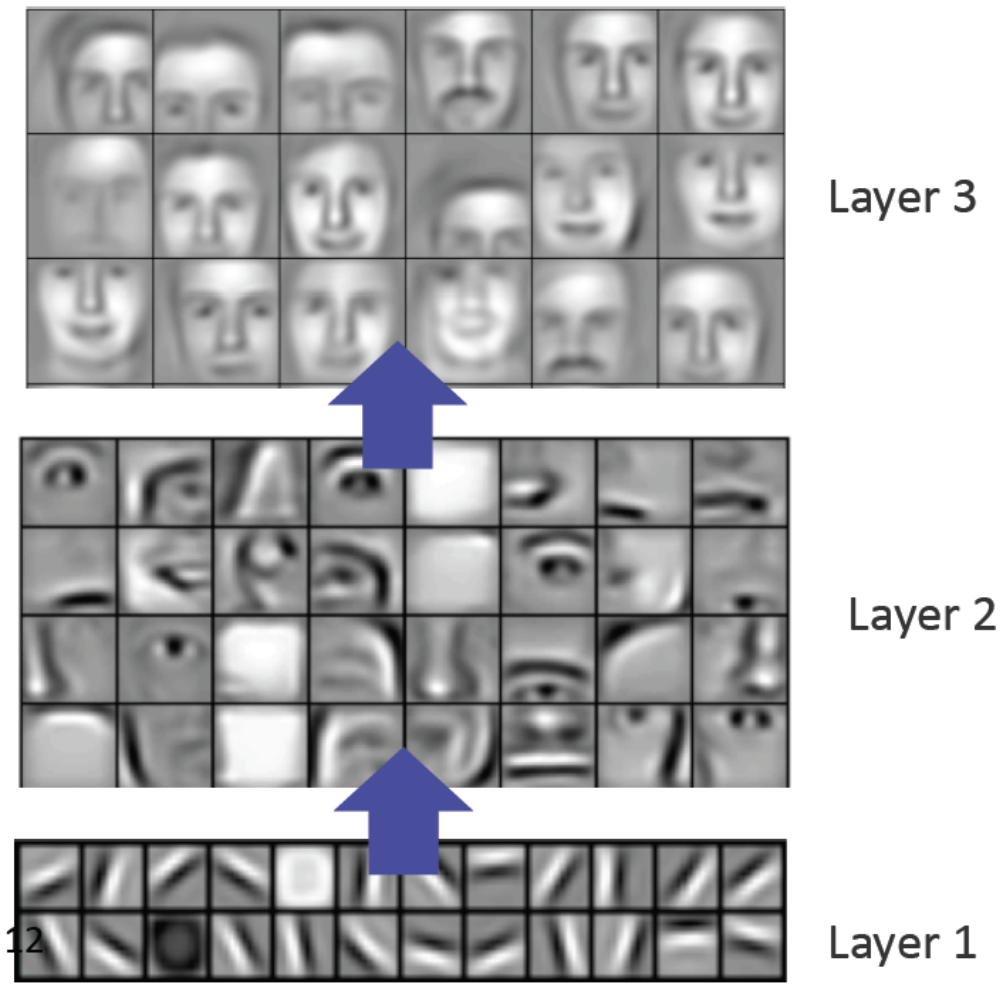
A family of methods that uses deep architectures to learn high-level feature representations（深層構造により抽象度が高い素性ベクトルを学習する手法）



例：深層学習から得られた素性 [Lee et al., 2009]

Input: Images (raw pixels)

→ Output: Features representing Edges, Body Parts, Full Faces



目次

1. Deep Learning の背景
   - ニューラルネットワーク（一層、二層）
   - 深層構造の可能性と限界
   - 2006 年のブレークスルー
2. 主な深層構造 (Deep Architecture)
   - Deep Belief Nets (DBN) [Hinton et al., 2006]
   - Stacked Auto-Encoders (SAE) [Bengio et al., 2006]
   - 機械学習の分野における Deep Learning の現状
3. 言語処理への応用
   - 非線形分類器としての応用 (Non-linear classifier)
   - 分散表現としての応用 (Distributed representation)
   - ディスカッション

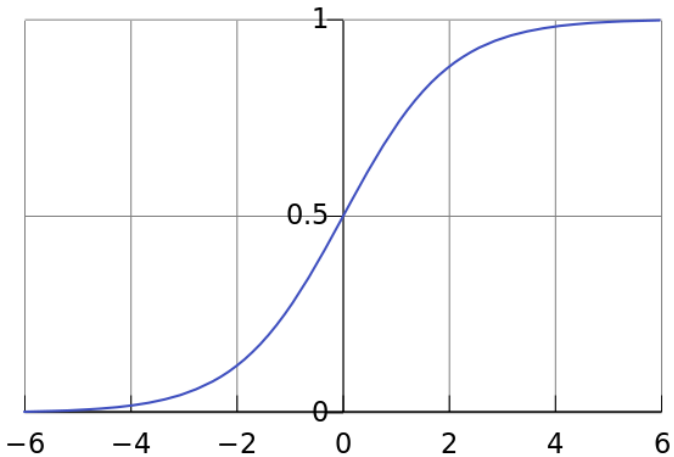
目次

機械学習の問題設定

- Training Data: a set of $(x^{(m)}, y^{(m)})_{m=\{1,2,..M\}}$ pairs
  - Input $x^{(m)} \in R^d$
  - Output $y^{(m)} = \{0, 1\}$
- Goal: Learn function $f : x \rightarrow y$ to predict correctly on new inputs x.
  - Step 1: Choose a function model family:
    - ★ e.g. logistic regression, support vector machines, neural networks
  - Step 2: Optimize parameters w on the Training Data
    - ★ e.g. minimize loss function $\min_w \sum_{m=1}^{M} (f_w(x^{(m)}) - y^{(m)})^2$

ロジスティック回帰（一層ネットワーク）

- Function model: $f(x) = \sigma(w^T \cdot x)$
  - Parameters: vector $w \in R^d$
  - σ is a non-linearity, e.g. sigmoid: $\sigma(z) = 1/(1 + \exp(-z))$



- Non-linearity will be important in expressiveness multi-layer nets. Other non-linearities, e.g., $tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$

ロジスティック回帰の学習：勾配

- Assume Squared-Error* $Loss(w) = \frac{1}{2}\sum_m (\sigma(w^T x^{(m)}) - y^{(m)})^2$
- Gradient: $\nabla_w Loss = \sum_m \left[\sigma(w^T x^{(m)}) - y^{(m)}\right] \sigma'(w^T x^{(m)}) x^{(m)}$
  - Define input into non-linearity $in^{(m)} = w^T x^{(m)}$
  - General form of gradient: $\sum_m Error^{(m)} * \sigma'(in^{(m)}) * x^{(m)}$
  - Derivative of sigmoid $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- Interpretation: Update $w = w - \gamma \nabla_w Loss$

$\sigma(w^T x^{(m)})$	$y^{(m)}$	*Error*	new w	new prediction
0	0	0	no change	0
1	1	0	no change	1
0	1	-1	$w + \gamma \sigma'(in^{(m)}) x^{(m)}$	≥ 0
1	0	+1	$w - \gamma \sigma'(in^{(m)}) x^{(m)}$	≤ 1

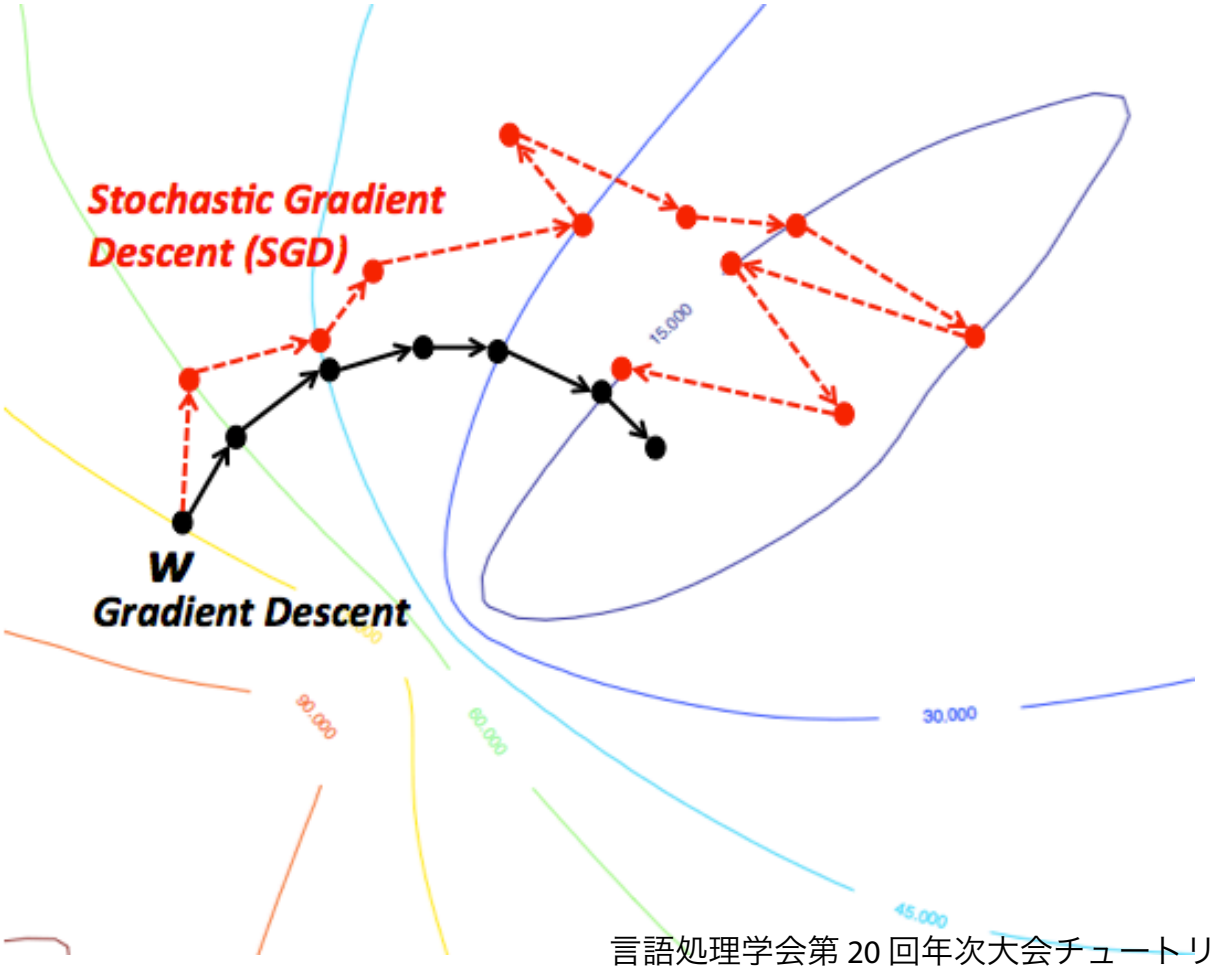
*An alternative is Cross-Entropy loss:
$\sum_m y^{(m)} \log(\sigma(w^T x^{(m)})) + (1 - y^{(m)}) \log(1 - \sigma(w^T x^{(m)}))$

ロジスティック回帰の学習：勾配降下法

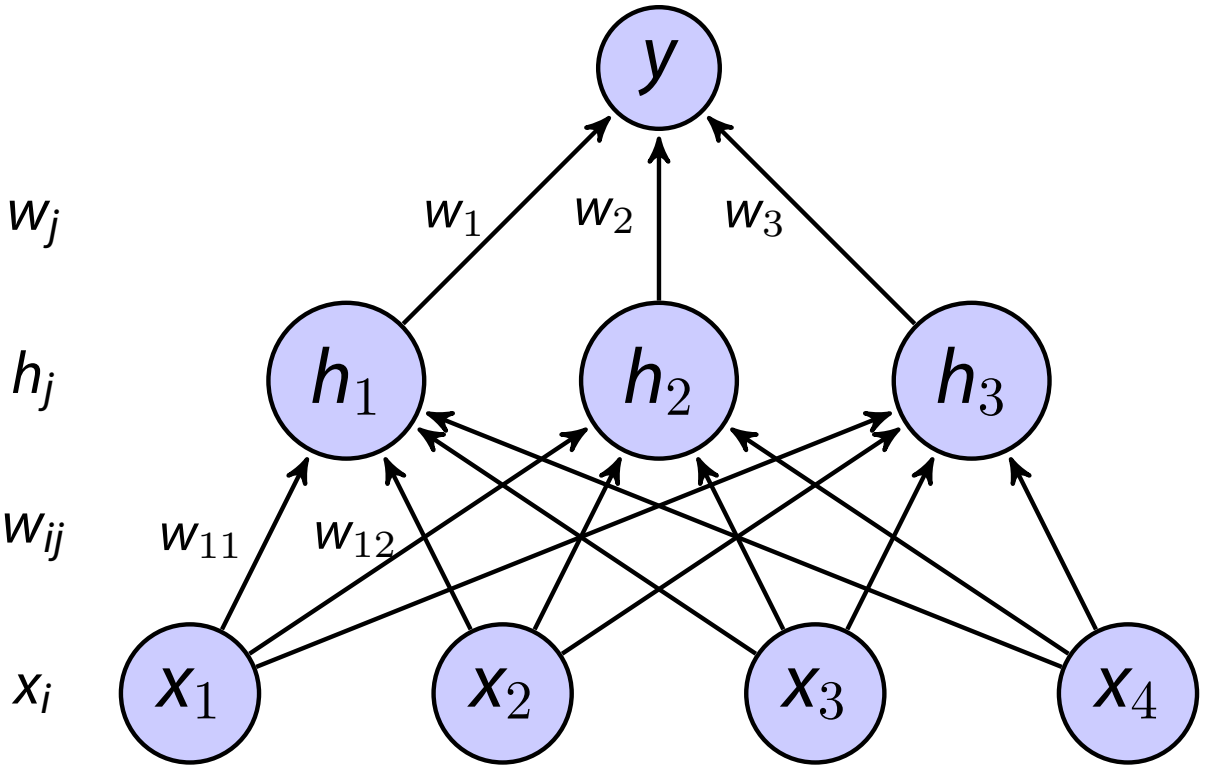
- General form of gradient: $\sum_m Error^{(m)} * \sigma'(in^{(m)}) * x^{(m)}$
- Gradient descent algorithm:
  1. Initialize w
  2. Compute $\nabla_w Loss = \sum_m Error^{(m)} * \sigma'(in^{(m)}) * x^{(m)}$
  3. $w \leftarrow w - \gamma(\nabla_w Loss)$
  4. Repeat steps 2-3 until some condition satisfied
- Stochastic gradient descent (SGD) algorithm:
  1. Initialize w
  2. for each sample $(x^{(m)}, y^{(m)})$ in training set:
  3. $w \leftarrow w - \gamma(Error^{(m)} * \sigma'(in^{(m)}) * x^{(m)})$
  4. Repeat loop 2-3 until some condition satisfied
- Learning rate $\gamma > 0$ & stopping condition are important in practice

SGD のイメージ図

- Loss objective contour plot: $\frac{1}{2}\sum_m(\sigma(w^T x^{(m)}) - y^{(m)})^2 + ||w||$
  - Gradient descent goes in steepest descent direction, but slower to compute per iteration for large datasets
  - SGD can be viewed as noisy descent, but faster per iteration
  - In practice, a good tradeoff is mini-batch SGD



二層ニューラルネットワーク



$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

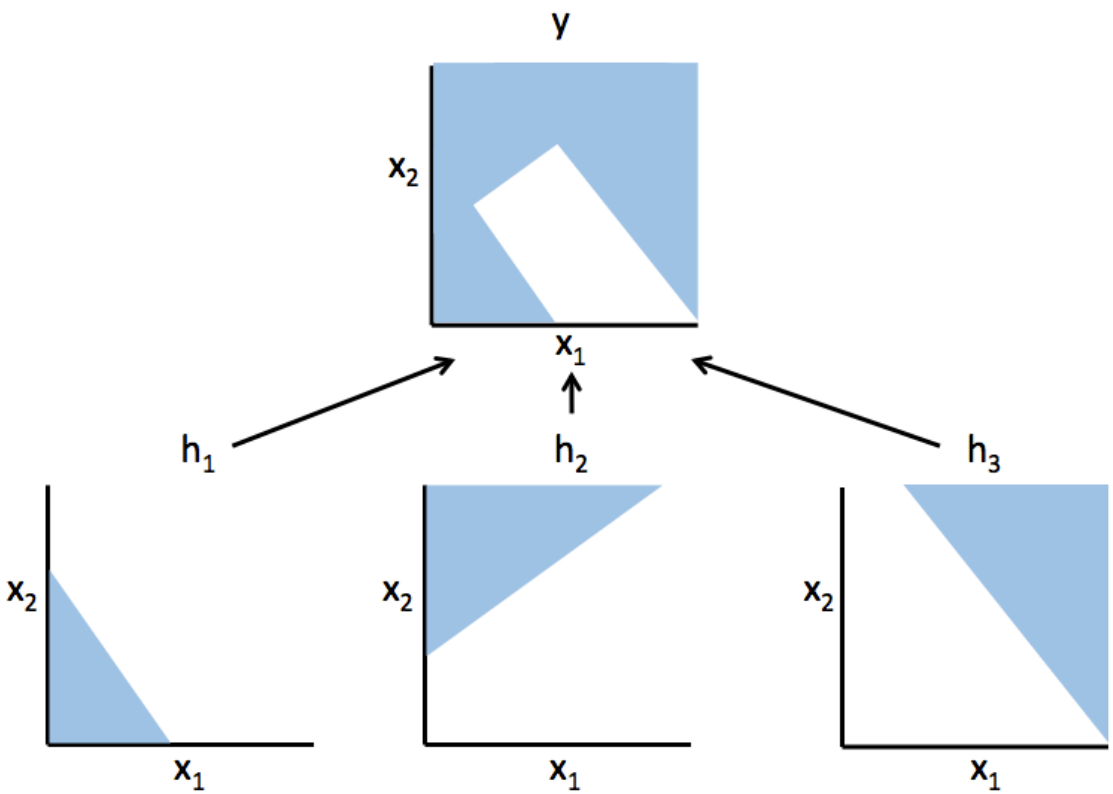
Hidden units h_j's can be viewed as new "features" from combining x_i's

---

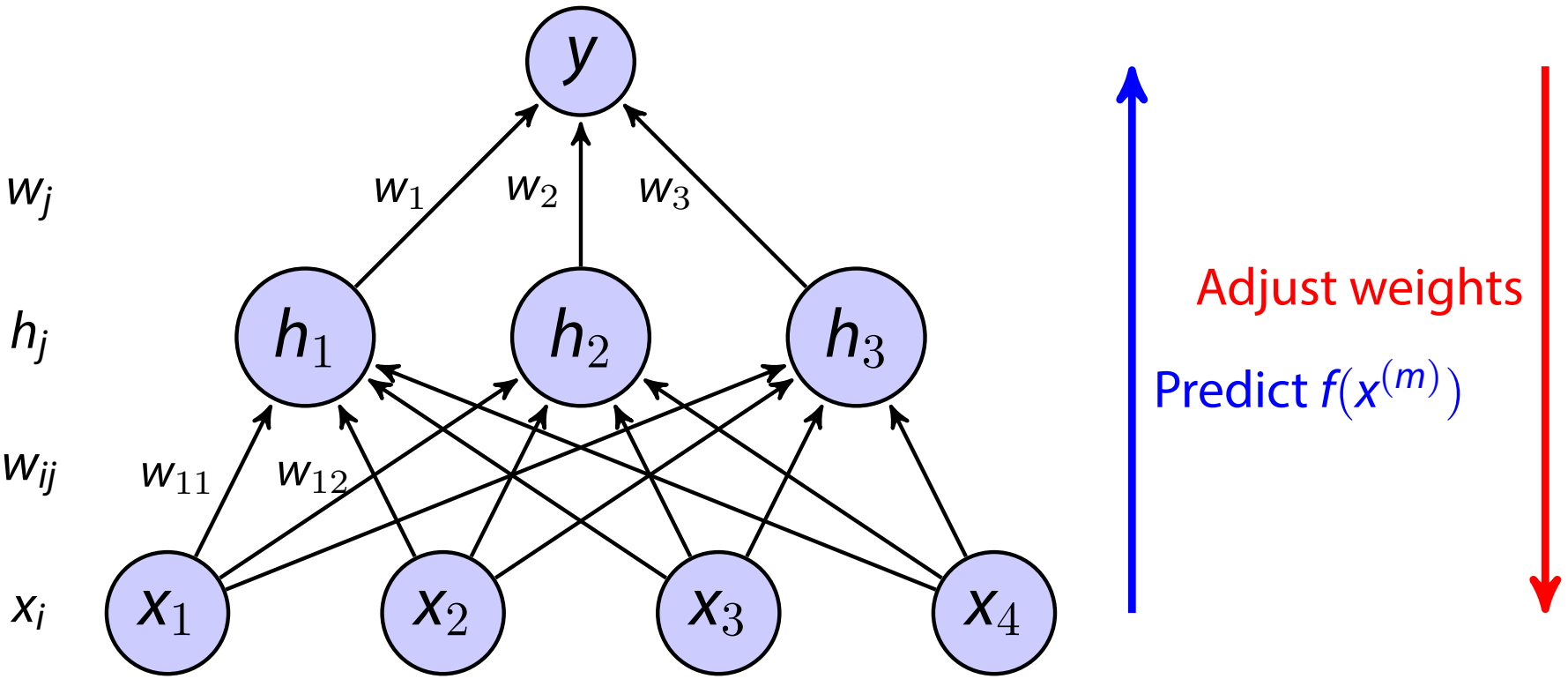
Called Multilayer Perceptron (MLP), but more like multilayer logistic regression

非線形モデルの強み

- A deeper architecture is more expressive than a shallow one given same number of nodes [Bishop, 1995]
  - 1-layer nets only model linear hyperplanes
  - 2-layer nets can model any continuous function (given sufficient nodes)
  - ＞3-layer nets can do so with fewer nodes



ニューラルネットワークの学習：誤差逆伝播



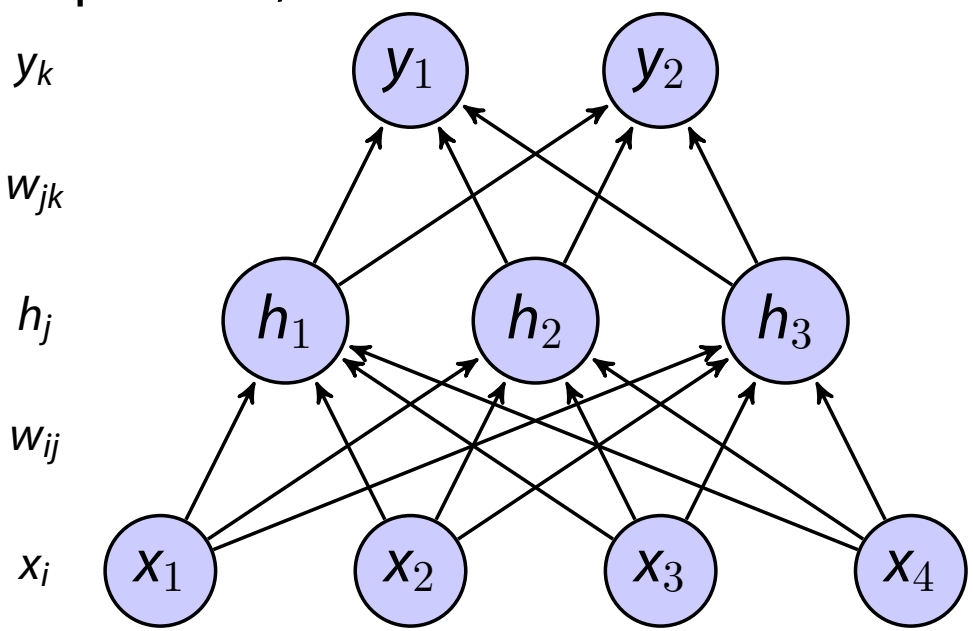
1. For each sample, compute $f(x^{(m)}) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i^{(m)}))$
2. If $f(x^{(m)}) \neq y^{(m)}$, back-propagate error and adjust weights $\{w_{ij}, w_j\}$.

ニューラルネットワークの学習：誤差逆伝播

All updates involve some scaled error from output $*$ input feature:

- $\frac{\partial Loss}{\partial w_{jk}} = \delta_k h_j$ where $\delta_k = [\sigma(in_k) - y_k]\,\sigma'(in_k)$
- $\frac{\partial Loss}{\partial w_{ij}} = \delta_j x_i$ where $\delta_j = \left[\sum_k \delta_k w_{jk}\right]\sigma'(in_j)$

After forward pass, compute δ_k from final layer, then δ_j for previous layer. For deeper nets, iterate backwards further.



目次

深層の可能性

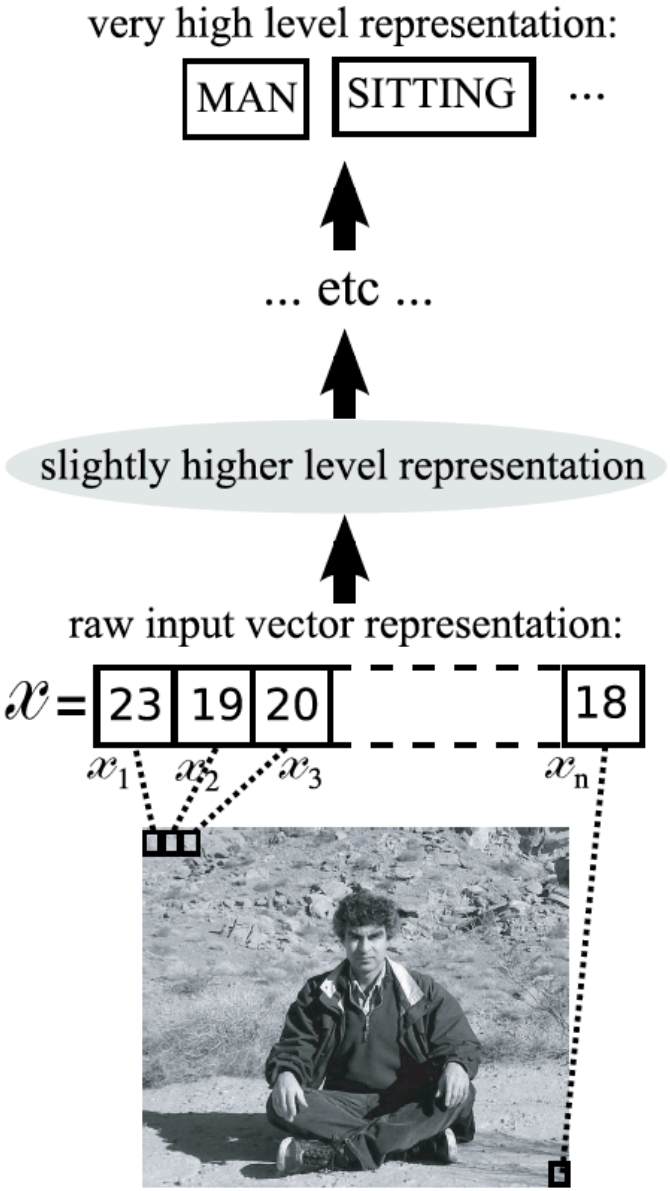


Figure from [Bengio, 2009]

- *Understanding in AI* requires high-level abstractions, modeled by highly non-linear functions
- These abstractions must disentangle factors of variation in data (e.g. 3D pose, lighting)
- Deep Architecture is one way to achieve this: each intermediate layer is a successively higher level abstraction

深層の可能性

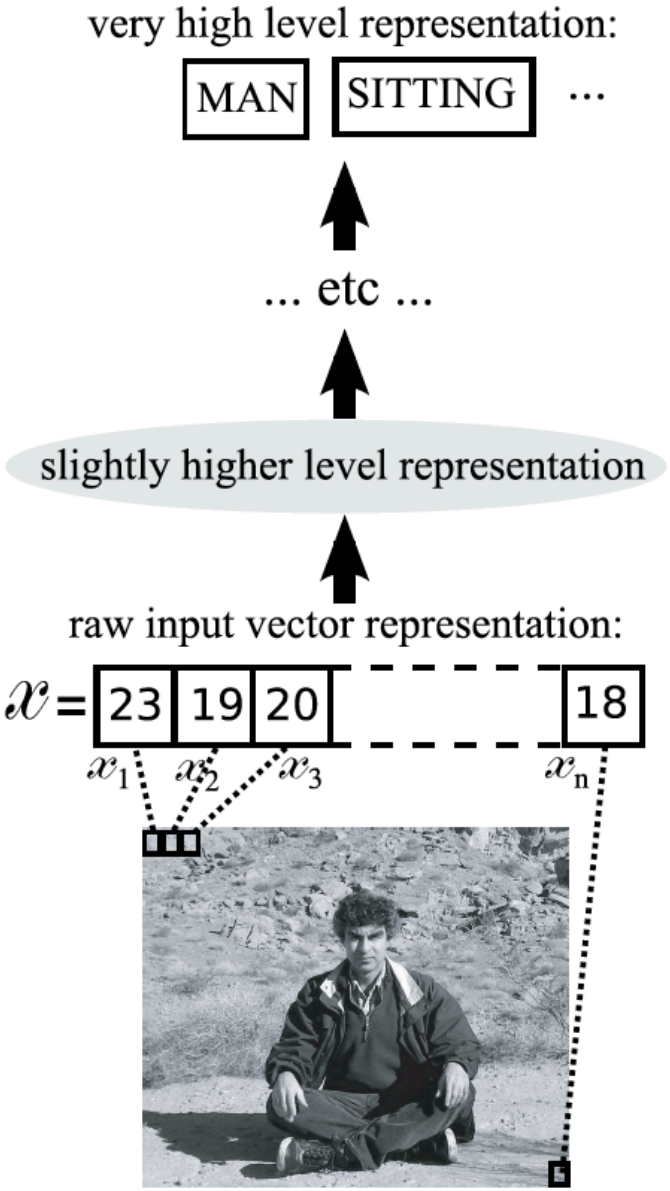
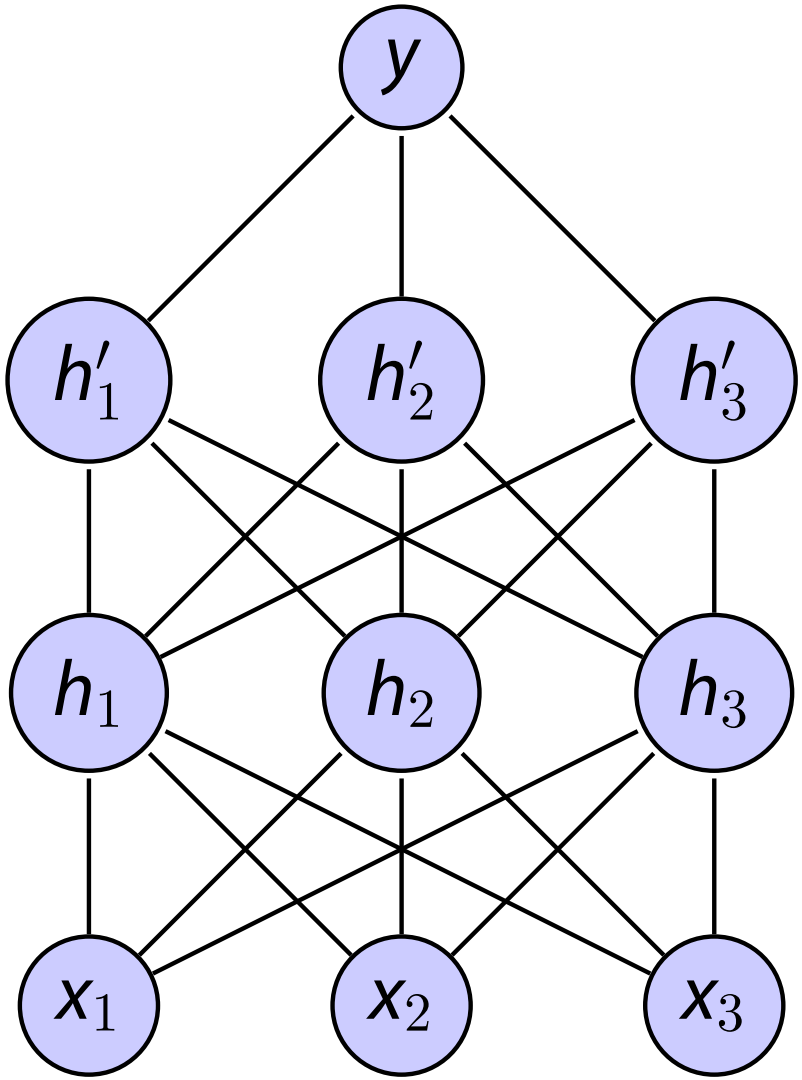


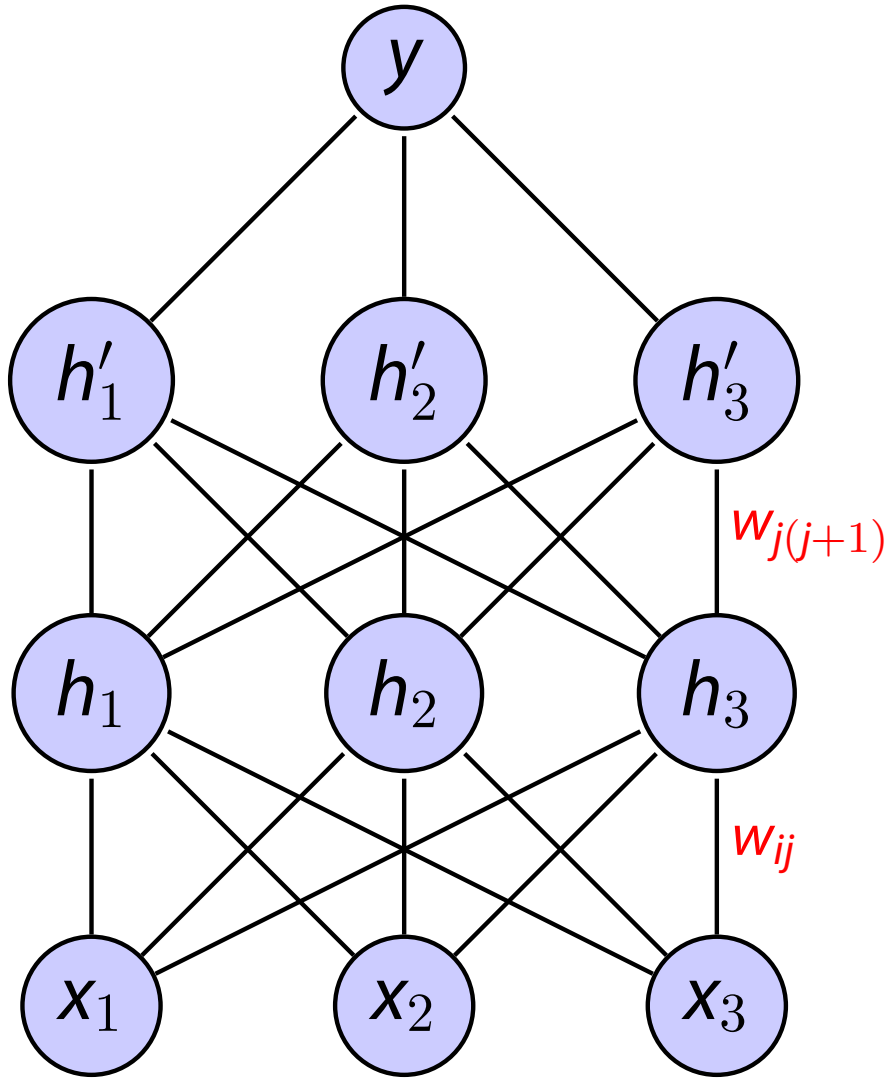
Figure from [Bengio, 2009]



深層構造による学習の難点

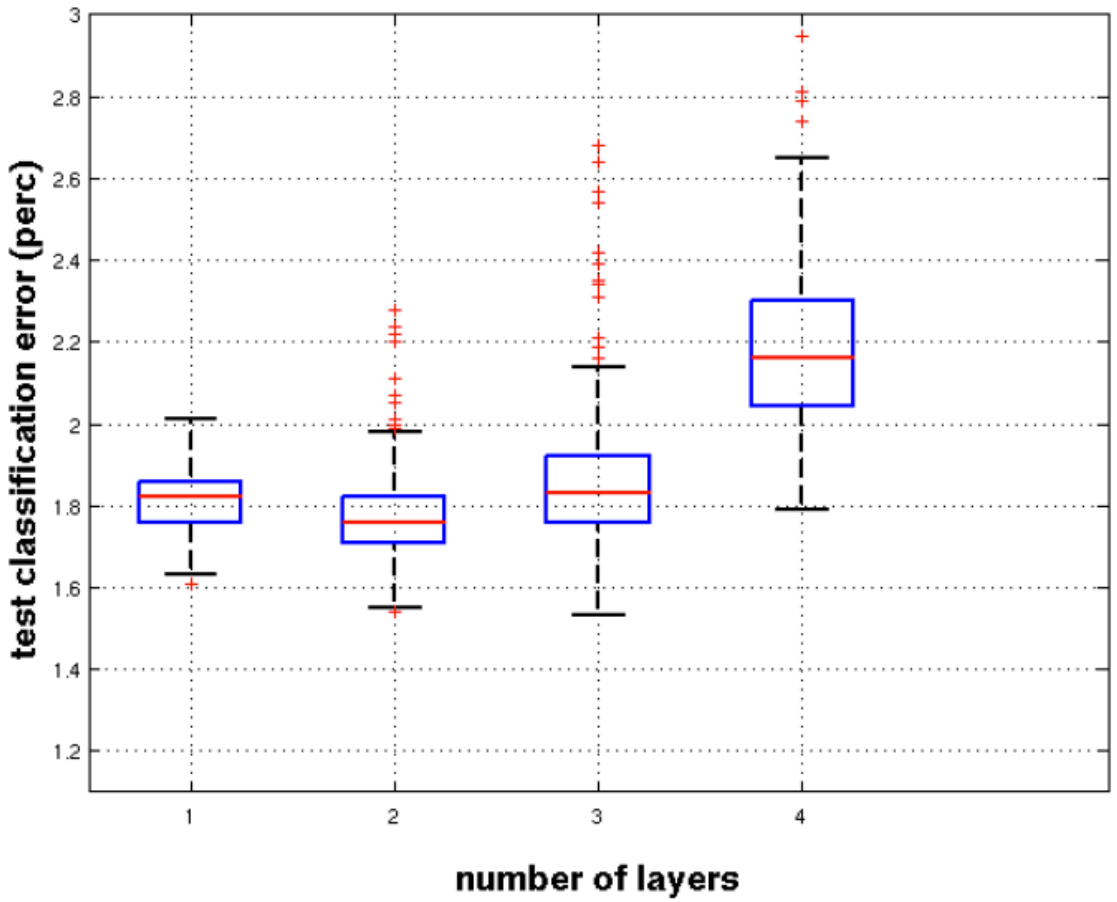
Vanishing gradient problem in Backpropagation

- $\frac{\partial Loss}{\partial w_{ij}} = \frac{\partial Loss}{\partial in_j}\frac{\partial in_j}{\partial w_{ij}} = \delta_j x_i$
- $\delta_j = \left[\sum_{j+1} \delta_{j+1} w_{j(j+1)}\right] \sigma'(in_j)$
- δ_j may vanish after repeated multiplication



深層構造による学習の限界 [Erhan et al., 2009]

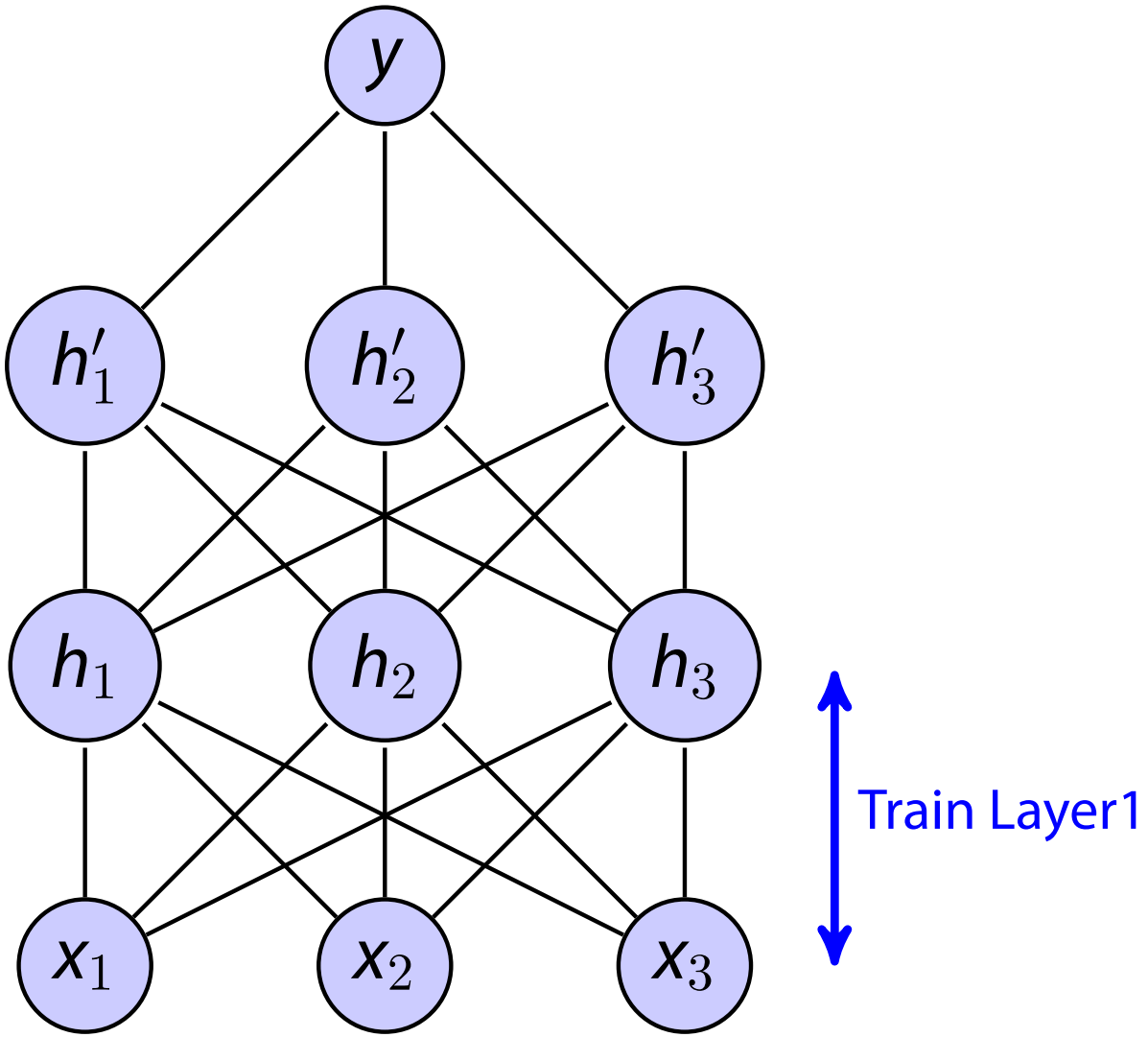
- MNIST digit classification task
- Train neural net by Backpropagation (from random initialization of w_{ij})
- On deep nets, we're apparently getting a local optimum that does not generalize well



目次

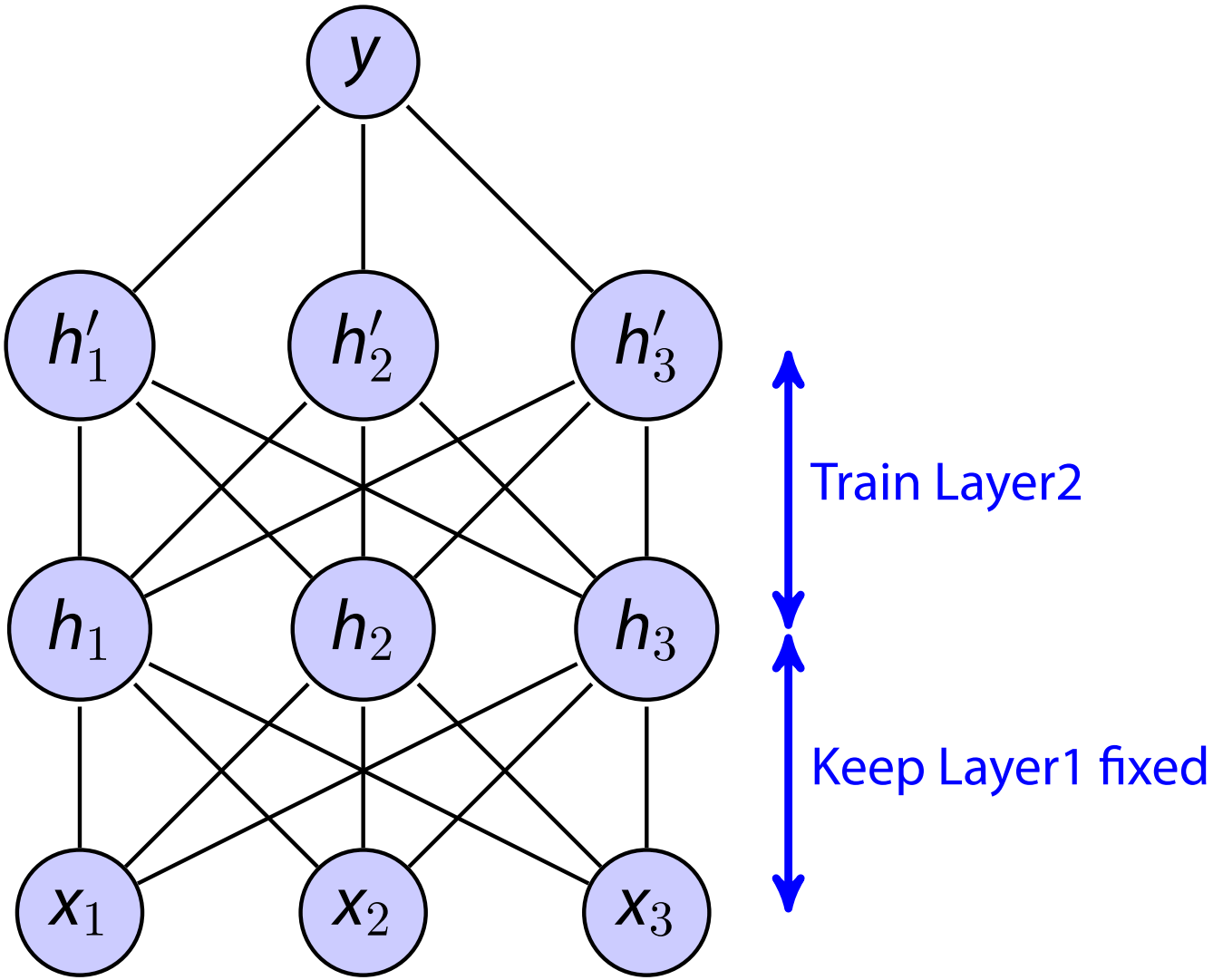
Layer-wise Pre-training [Hinton et al., 2006]

First, train one layer at a time, optimizing data-likelihood objective $P(x)$



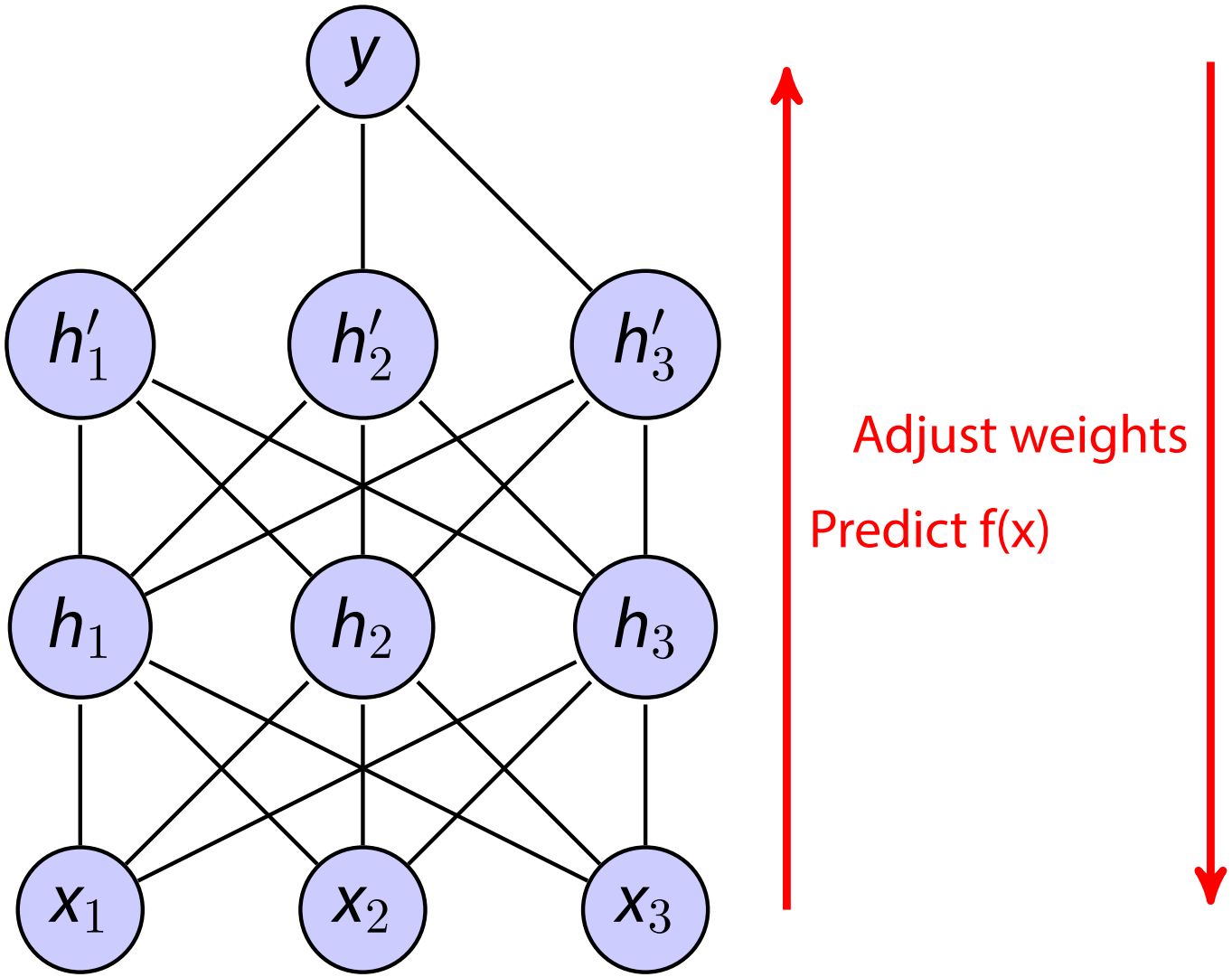
Layer-wise Pre-training [Hinton et al., 2006]

First, train one layer at a time, optimizing data-likelihood objective $P(x)$



Layer-wise Pre-training [Hinton et al., 2006]

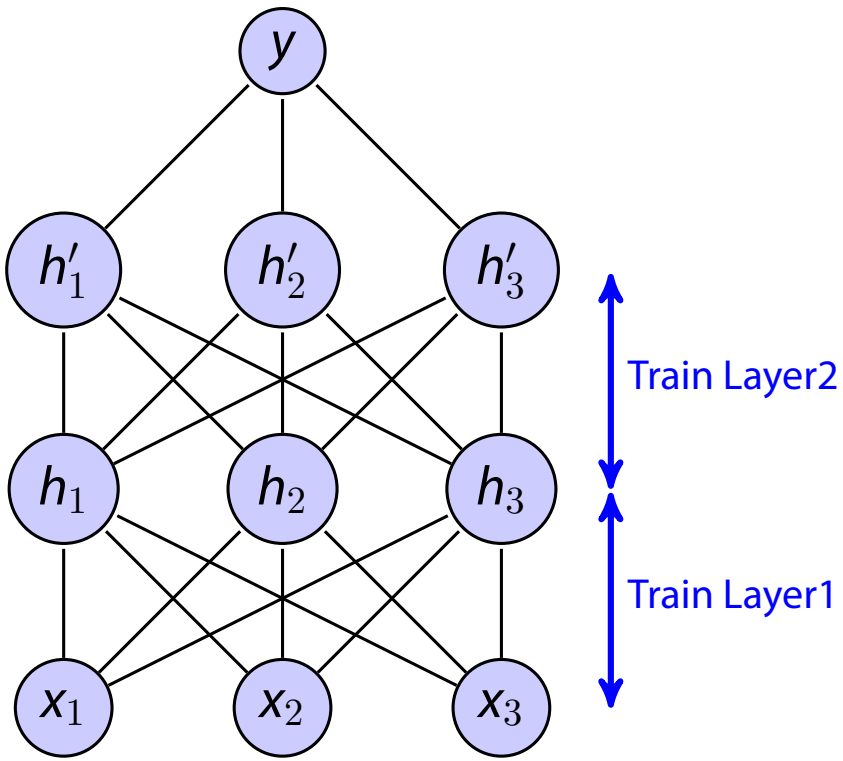
Finally, fine-tune labeled objective $P(y|x)$ by Backpropagation



Layer-wise Pre-training [Hinton et al., 2006]

Key Idea:
Focus on modeling the input $P(X)$ better with each successive layer.
Worry about optimizing the task $P(Y|X)$ later.

> *"If you want to do computer vision, first learn computer graphics." – Geoff Hinton*



Extra advantage: Can exploit large amounts of unlabeled data!

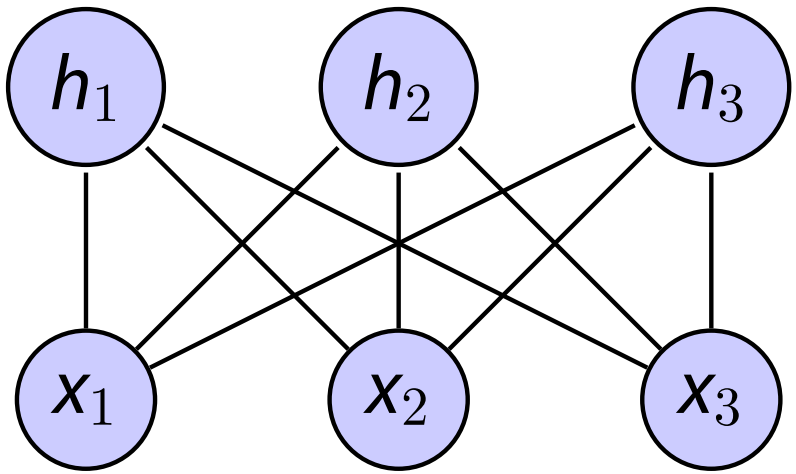
目次

Deep Learning のパラダイム

- Recall the problem setup: Learn function $f : x \rightarrow y$
- But rather doing this directly, we first learn hidden features h that model input x, i.e. $x \rightarrow h \rightarrow y$
- How do we discover useful latent features h from data x?
  - Different Deep Learning methods differ by this basic component
  - e.g. Deep Belief Nets use Restricted Boltzmann Machines (RBMs)

Restricted Boltzmann Machine (RBM)

- RBM is a simple energy-based model: $p(x,h) = \frac{1}{Z_\theta} \exp\left(-E_\theta(x,h)\right)$
  - with only h-x interactions: $E_\theta(x,h) = -x^T W h - b^T x - d^T h$
  - here, we assume h_j and x_i are binary variables
  - normalizer: $Z_\theta = \sum_{(x,h)} \exp(-E_\theta(x,h))$ is called partition function



- Example:
  - Let weights (h_1, x_1), (h_1, x_3) be positive, others be zero, $b = d = 0$.
  - Then this RBM defines a distribution over $[x_1, x_2, x_3, h_1, h_2, h_3]$ where $p(x_1 = 1, x_2 = 0, x_3 = 1, h_1 = 1, h_2 = 0, h_3 = 0)$ has high probability

RBM 事後分布の計算（簡単）

- Computing $p(h|x)$ is easy due to factorization:

$$
\begin{aligned}
p(h|x) &= \frac{p(x,h)}{\sum_h p(x,h)} = \frac{1/Z_\theta \exp(-\mathrm{E}(\mathrm{x},\mathrm{h}))}{\sum_h 1/Z_\theta \exp(-\mathrm{E}(\mathrm{x},\mathrm{h}))} \\
&= \frac{\exp(x^T W h + b^T x + d^T h)}{\sum_h \exp(x^T W h + b^T x + d^T h)} \\
&= \frac{\prod_j \exp(x^T W_j h_j + d_j h_j) \cdot \exp(b^T x)}{\sum_{h_1 \in \{0,1\}} \sum_{h_2 \in \{0,1\}} \cdots \sum_{h_j} \prod_j \exp(x^T W_j h_j + d_j h_j) \cdot \exp(b^T x)} \\
&= \frac{\prod_j \exp(x^T W_j h_j + d_j h_j)}{\prod_j \sum_{h_j \in \{0,1\}} \exp(x^T W_j h_j + d_j h_j)} \\
&= \prod_j \frac{\exp(x^T W_j h_j + d_j h_j)}{\sum_{h_j \in \{0,1\}} \exp(x^T W_j h_j + d_j h_j)} = \prod_j p(h_j|x)
\end{aligned}
$$

- Note $p(h_j = 1|x) = \exp(x^T W_j + d_j)/Z = \sigma(x^T W_j + d_j)$
- Similarly, computing $p(x|h) = \prod_i p(x_i|h)$ is easy

RBM の学習：最尤推定（大変）

Derivative of the Log-Likelihood: $\partial_{w_{ij}} \log P_W(x = x^{(m)})$

$$= \partial_{w_{ij}} \log \sum_h P_W(x = x^{(m)}, h) \tag{1}$$

$$= \partial_{w_{ij}} \log \sum_h \frac{1}{Z_W} \exp\left(-E_w(x^{(m)}, h)\right) \tag{2}$$

$$= -\partial_{w_{ij}} \log Z_W + \partial_{w_{ij}} \log \sum_h \exp\left(-E_w(x^{(m)}, h)\right) \tag{3}$$

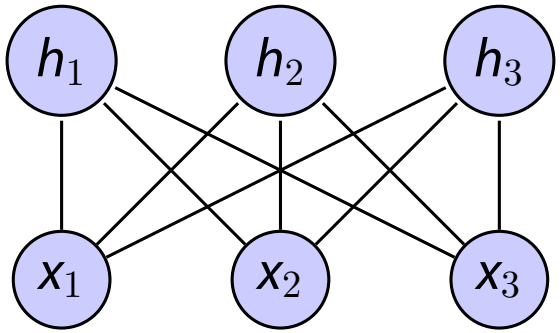
$$= \frac{1}{Z_W} \sum_{h,x} e^{(-E_w(x,h))} \partial_{w_{ij}} E_w(x, h) - \frac{1}{\sum_h e^{(-E_w(x^{(m)},h))}} \sum_h e^{(-E_w(x^{(m)},h))} \partial_{w_{ij}} E_w(x^{(m)}, h)$$

$$= \sum_{h,x} P_W(x, h)[\partial_{w_{ij}} E_w(x, h)] - \sum_h P_W(x^{(m)}, h)[\partial_{w_{ij}} E_w(x^{(m)}, h)] \tag{4}$$

$$= -\mathbb{E}_{p(x,h)}[x_i \cdot h_j] + \mathbb{E}_{p(h|x=x^{(m)})}[x_i^{(m)} \cdot h_j] \tag{5}$$

Second term (positive phase) increases probability of $x^{(m)}$; First term (negative phase) decreases probability of samples generated by the model

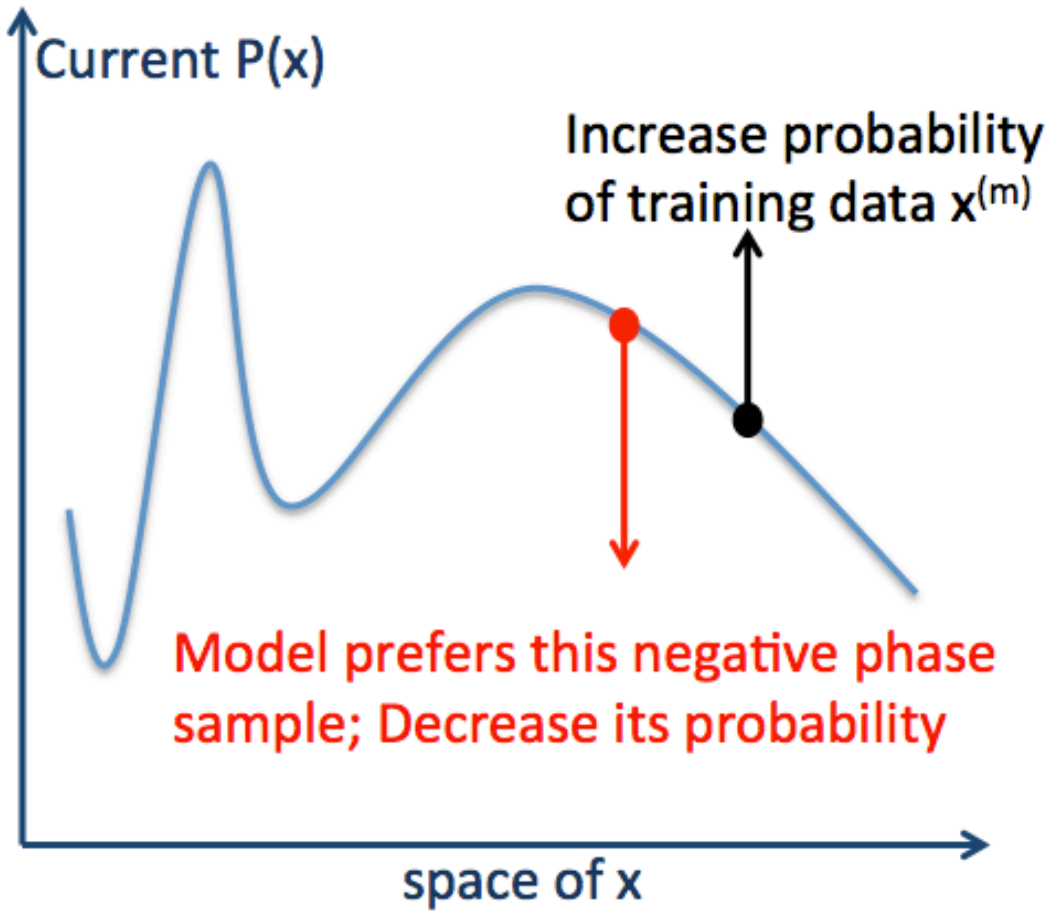
Contrastive Divergence Algorithm



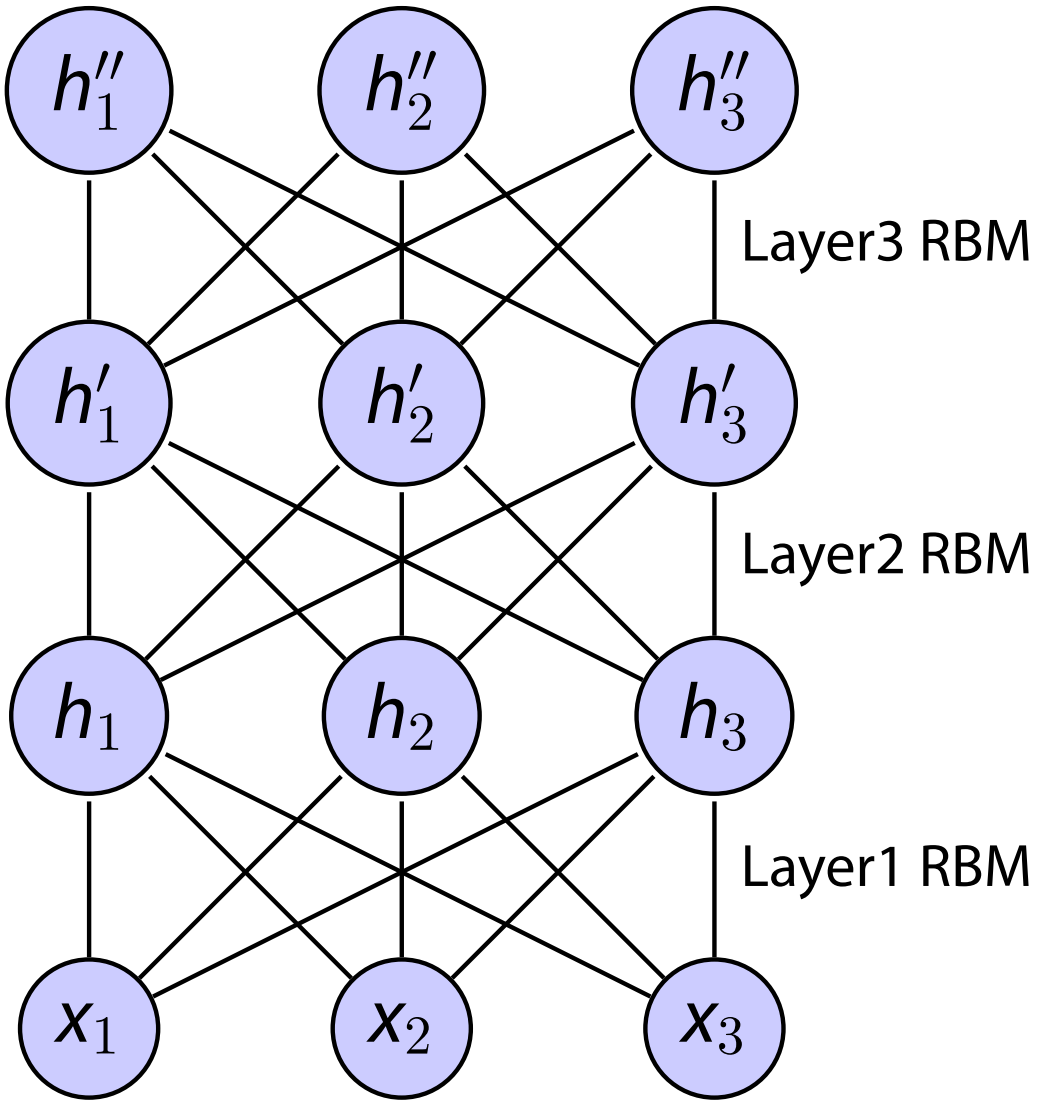
- The negative phase term ($\mathbb{E}_{p(x,h)}[x_i \cdot h_j]$) is expensive because it requires sampling (x,h) from the model
- Gibbs Sampling (sample x then h iteratively) works, but waiting for convergence at each gradient step is slow.
- Contrastive Divergence is a faster but biased method: initialize with training point and wait only a few (usu. 1) sampling steps
  1. Let $x^{(m)}$ be training point, $W = [w_{ij}]$ be current model weights
  2. Sample $\hat{h}_j \in \{0, 1\}$ from $p(h_j | x = x^{(m)}) = \sigma(\sum_i w_{ij} x_i^{(m)} + d_j)\ \forall j$.
  3. Sample $\tilde{x}_i \in \{0, 1\}$ from $p(x_i | h = \hat{h}) = \sigma(\sum_j w_{ij} \hat{h}_j + b_i)\ \forall i$.
  4. Sample $\tilde{h}_j \in \{0, 1\}$ from $p(h_j | x = \tilde{x}) = \sigma(\sum_i w_{ij} \tilde{x}_i + d_j)\ \forall j$.
  5. $w_{ij} \leftarrow w_{ij} + \gamma(x_i^{(m)} \cdot \hat{h}_j - \tilde{x}_i \cdot \tilde{h}_j)$

Contrastive Divergence のイメージ図

- Goal: Make RBM $p(x, h)$ have high probability on training samples
- To do so, we'll "steal" probability mass from nearby samples that incorrectly preferred by the model
- For detailed analysis, see [Carreira-Perpinan and Hinton, 2005]



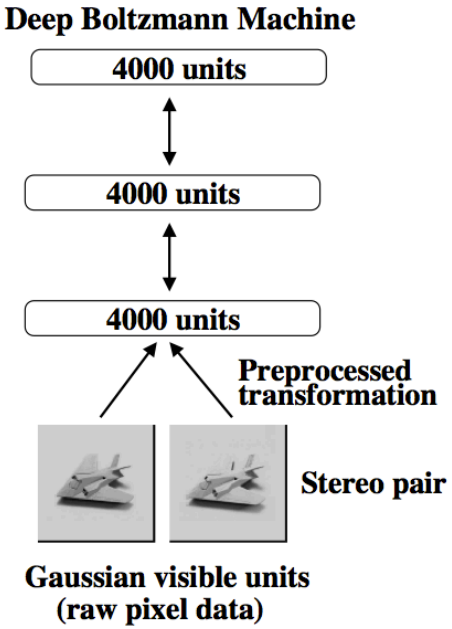
Deep Belief Nets (DBN) = RBM の積み重ね

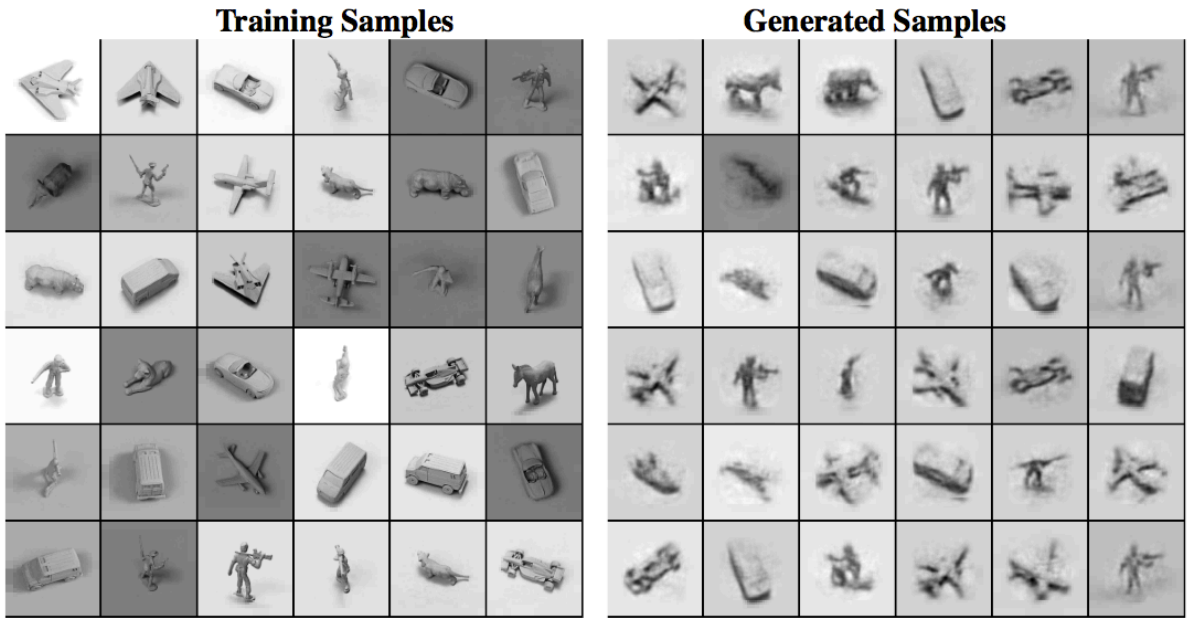


- DBN defines a probabilistic generative model $p(x) = \sum_{h,h',h''} p(x|h)p(h|h')p(h',h'')$ (top 2 layers is interpreted as a RBM; lower layers are directed sigmoids)
- Stacked RBMs can also be used to initialize a Deep Neural Network (DNN)

Deep Generative Model によるデータ生成

After training on 20k images, the generative model of [Salakhutdinov and Hinton, 2009]* can generate random images (dimension=8976) that are amazingly realistic!





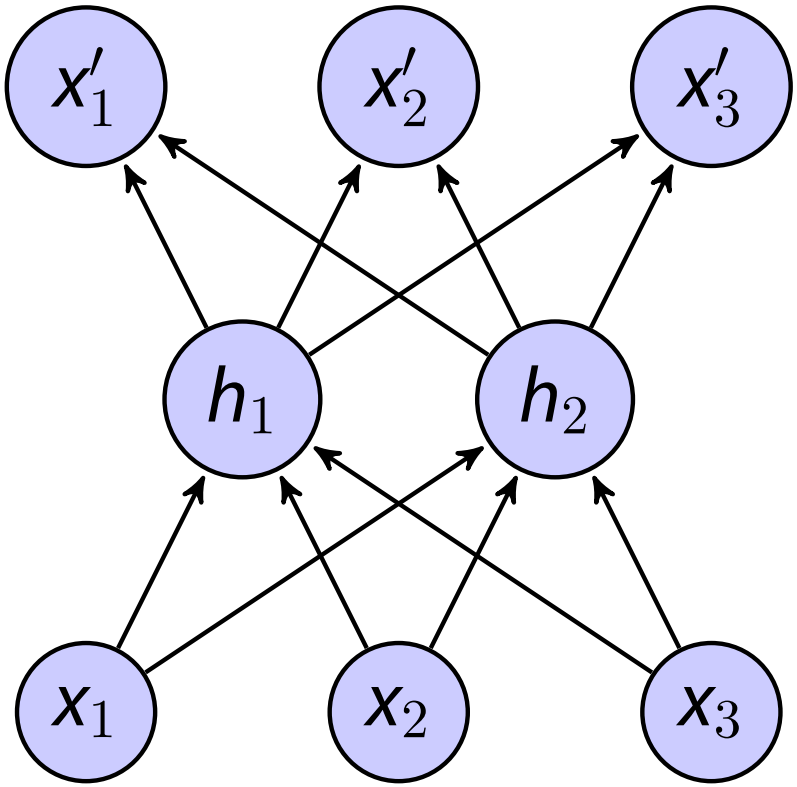
This model is a Deep Boltzmann Machine (DBM), different from Deep Belief Nets (DBN) but also built by stacking RBMs.

Deep Belief Nets (DBN)：まとめ

1. Layer-wise pre-training is the innovation that rekindled interest in deep architectures.
2. Pre-training focuses on optimizing likelihood on the data, not the target label. First model $p(x)$ to do better $p(y|x)$.
3. Why RBM? $p(h|x)$ is tractable, so it's easy to stack.
4. RBM training can be expensive. Solution: contrastive divergence
5. DBN formed by stacking RBMs is a probabilistic generative model

目次

Auto-Encoders: RBM の代わり



Decoder: $x' = \sigma(W'h + d)$

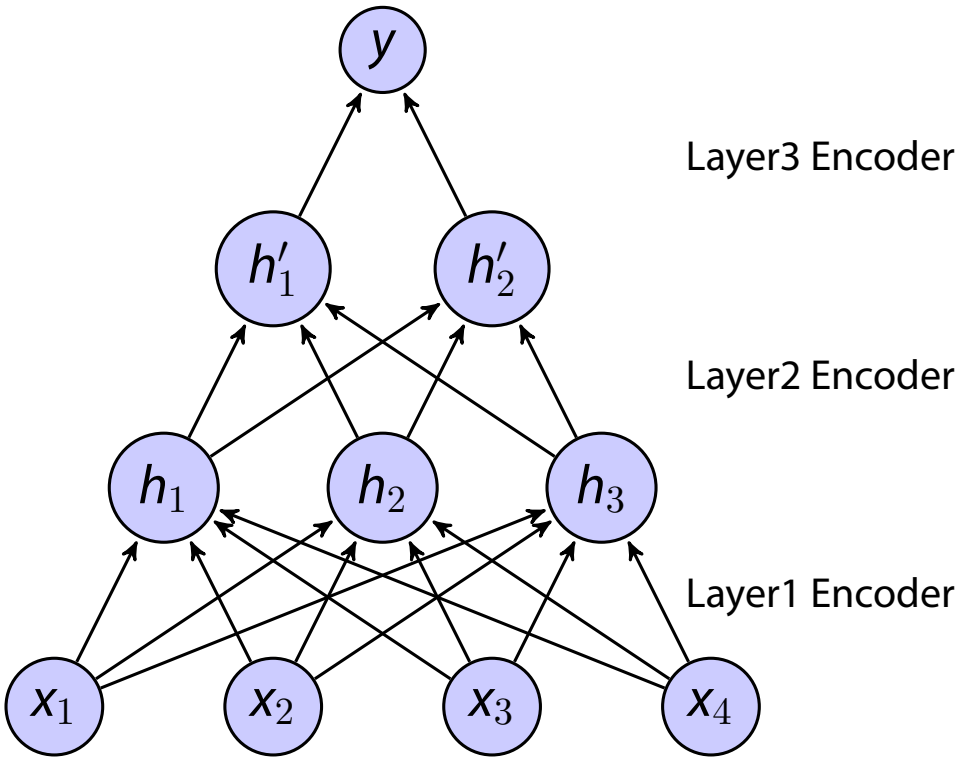
Encoder: $h = \sigma(Wx + b)$

Encourage h to give small reconstruction error:

- e.g. $Loss = \sum_m ||x^{(m)} - DECODER(ENCODER(x^{(m)}))||^2$
- Reconstruction: $x' = \sigma(W'\sigma(Wx + b) + d)$
- This can be trained with the same Backpropagation algorithm for 2-layer nets, with $x^{(m)}$ as both input and output

Stacked Auto-Encoders (SAE)

- The encoder/decoder gives same form $p(h|x)$, $p(x|h)$ as RBMs, so can be stacked in the same way to form Deep Architectures

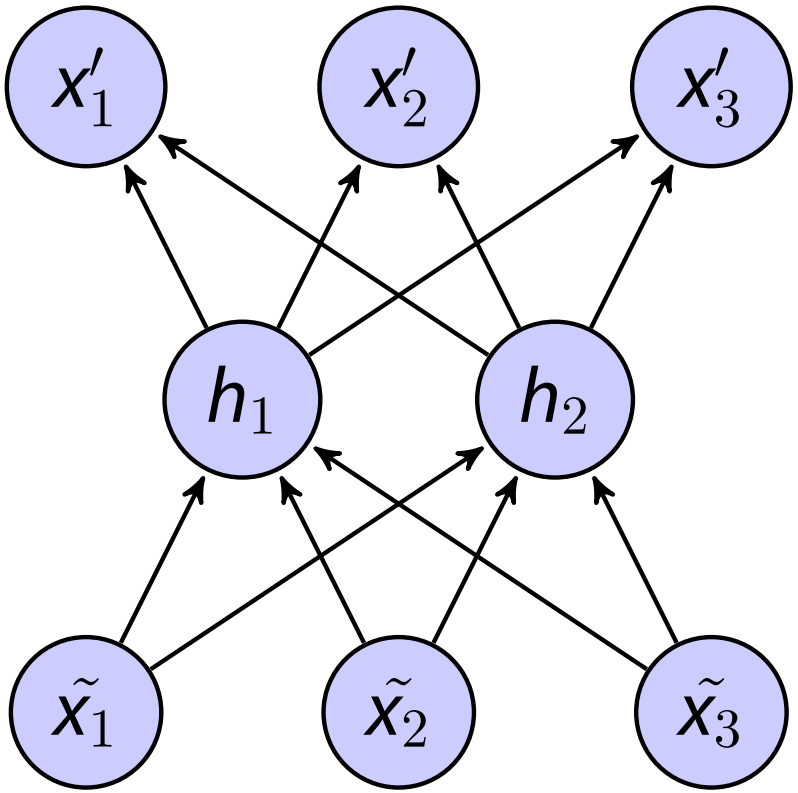


- Unlike RBMs, Auto-encoders are deterministic.
  - $h = \sigma(Wx + b)$, not $p(h = \{0, 1\}) = \sigma(Wx + b)$
  - Disadvantage: Can't form deep generative model
  - Advantage: Fast to train, and useful still for Deep Neural Nets

様々な Auto-Encoder

- Enforce compression to get latent factors (lower dimensional *h*)
- Linear encoder/decoder with squared reconstruction error learns same subspace of PCA [Bourlard and Kamp, 1988]
- Enforce sparsity and over-complete representations (high dimensional *h*) [Ranzato et al., 2006]
- Incorporate domain knowledge, e.g. denoising auto-encoders [Vincent et al., 2010]

Denoising Auto-Encoders



Decoder: $x' = \sigma(W'h + d)$

Encoder: $h = \sigma(W\tilde{x} + b)$

$\tilde{x} = x +$ noise

1. Perturb input data x to $\tilde{x}$ using invariance from domain knowledge.
2. Train weights to reduce reconstruction error with respect to original input: $||x - x'||$

Denoising Auto-Encoders

- Example: Randomly shift, rotate, and scale input image; add Gaussian or salt-and-pepper noise.
- A ”2” is a ”2” no matter how you add noise, so the auto-encoder will be forced to cancel the variations that are not important.

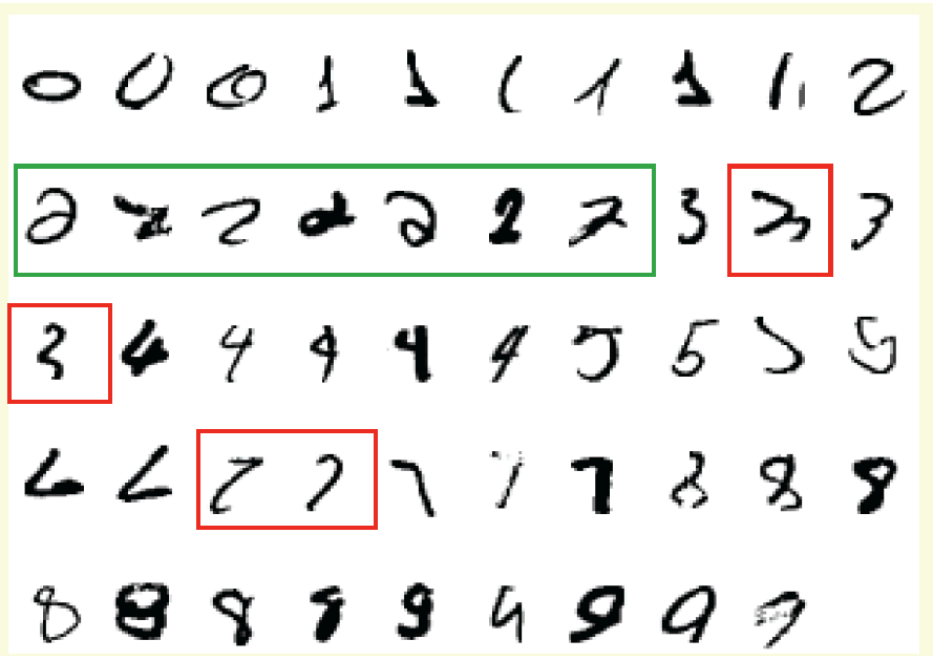


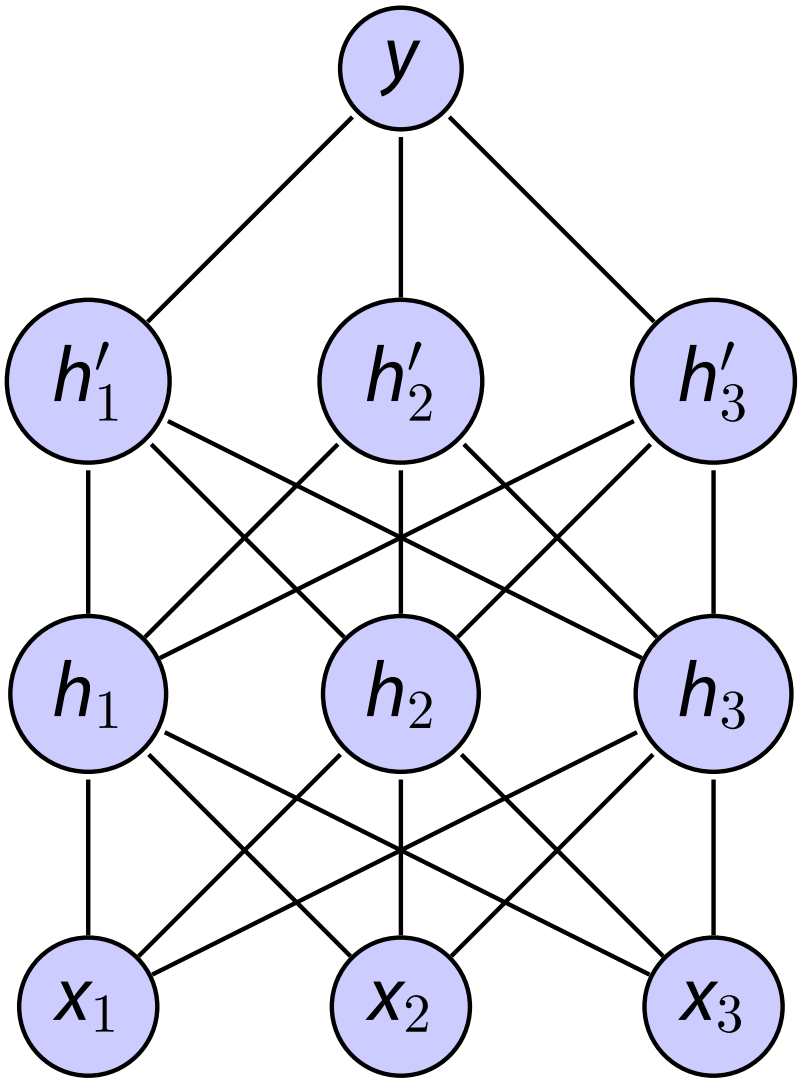
Figure from Geoff Hinton’s 2012 Coursera course, lecture 1: `https://www.coursera.org/course/neuralnets`

Stacked Auto-Encoders (SAE): まとめ

1. Auto-Encoders are cheaper alternatives to RBMs.
   - Not probabilistic, but fast to train using Backpropagation
2. Auto-Encoders learn to ”compress” and ”re-construct” input data. Again, the focus is on modeling $p(x)$ first.
3. Many variants, some provide ways to incorporate domain knowledge.

目次

Pre-Training はなぜ効く？[Erhan et al., 2010]



- A deep net can fit the training data in many ways (non-convex):
  1. By optimizing upper-layers really hard
  2. By optimizing lower-layers really hard
- Top-down vs. Bottom-up information
  1. Even if lower-layers are random weights, upper-layer may still fit well. But may not generalize to new data
  2. Pre-training with objective on $P(x)$ learns more generalizable features
- Pre-training seems to help put weights at a better local optimum

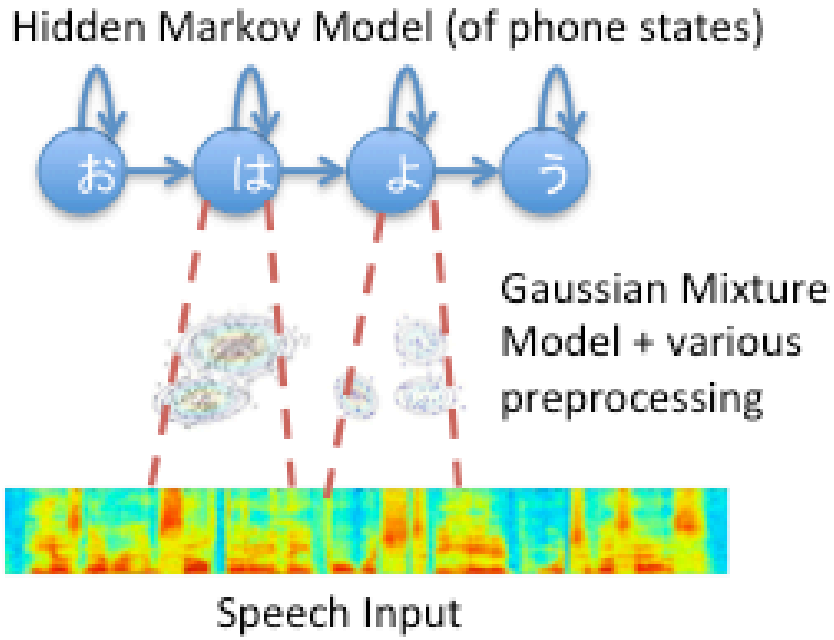
Pre-Training は本当に必要？

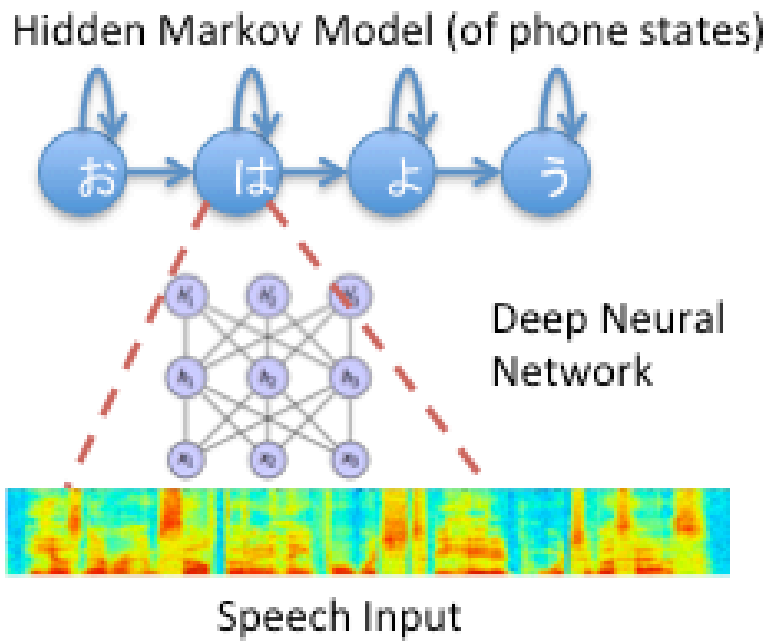
Answer in 2006: Yes!
Answer in 2014: No!

1. If initialization is done well by design (e.g. sparse connections and convolutional nets), maybe won't have vanishing gradient problem
2. If you have an extremely large datasets, maybe won't overfit. (But maybe that also means you want an ever deeper net)
3. New architectures are emerging: e.g. Stacked SVM's with random projections [Vinyals et al., 2012]

成功例：音声認識

Hybrid DNN-HMM system: (typically 3-8 layers, 2000 units/layer, 15 frames of input, 6000 output)





成功例：音声認識

Word Error Rate Results [Hinton et al., 2012a]:

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

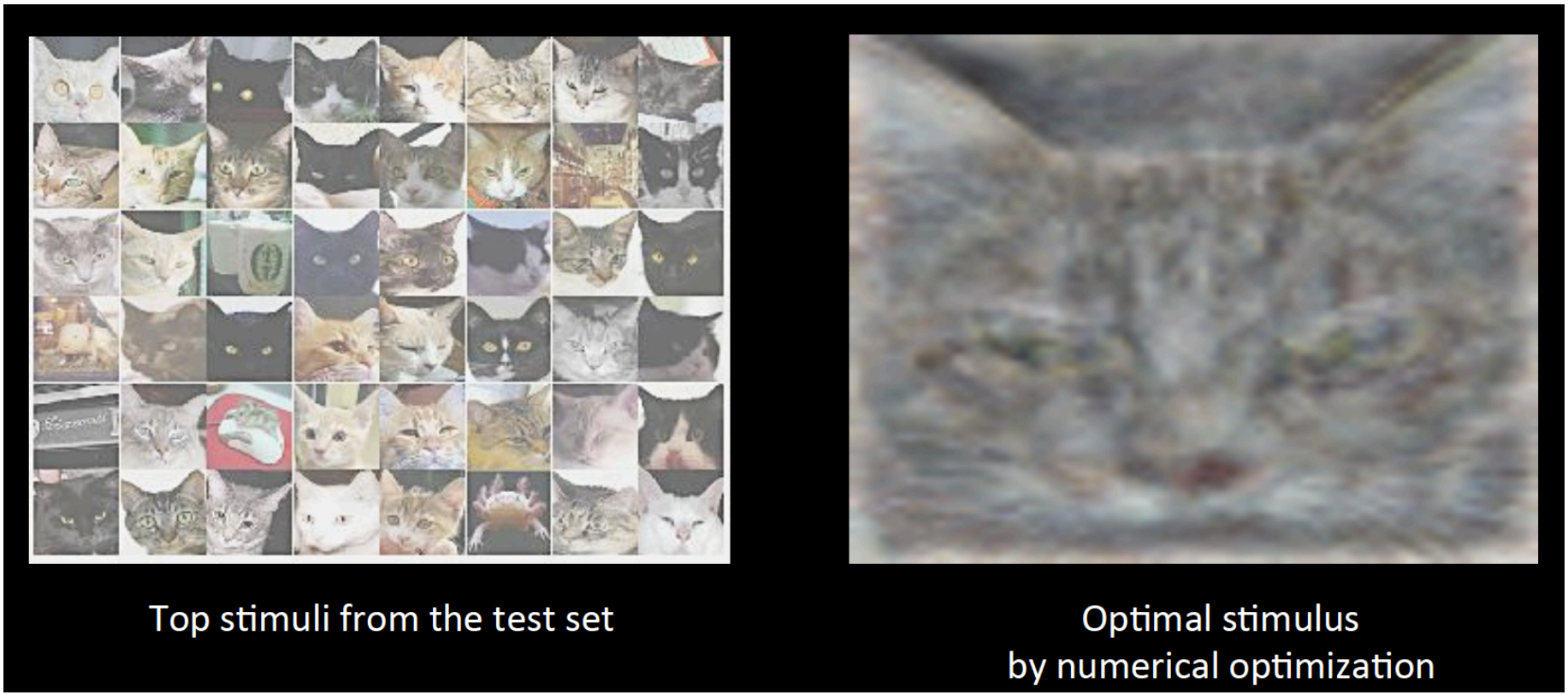
成功例：画像認識 [Le et al., 2012]

ImageNet Test Accuracy (22K categories):

Method	Accuracy
Random	0.005%
Previous State-of-the-art	9.3%
”9”-layer net, back-propagation without pre-training	13.6%
+ pre-training on 10 million Youtube images	15.8%

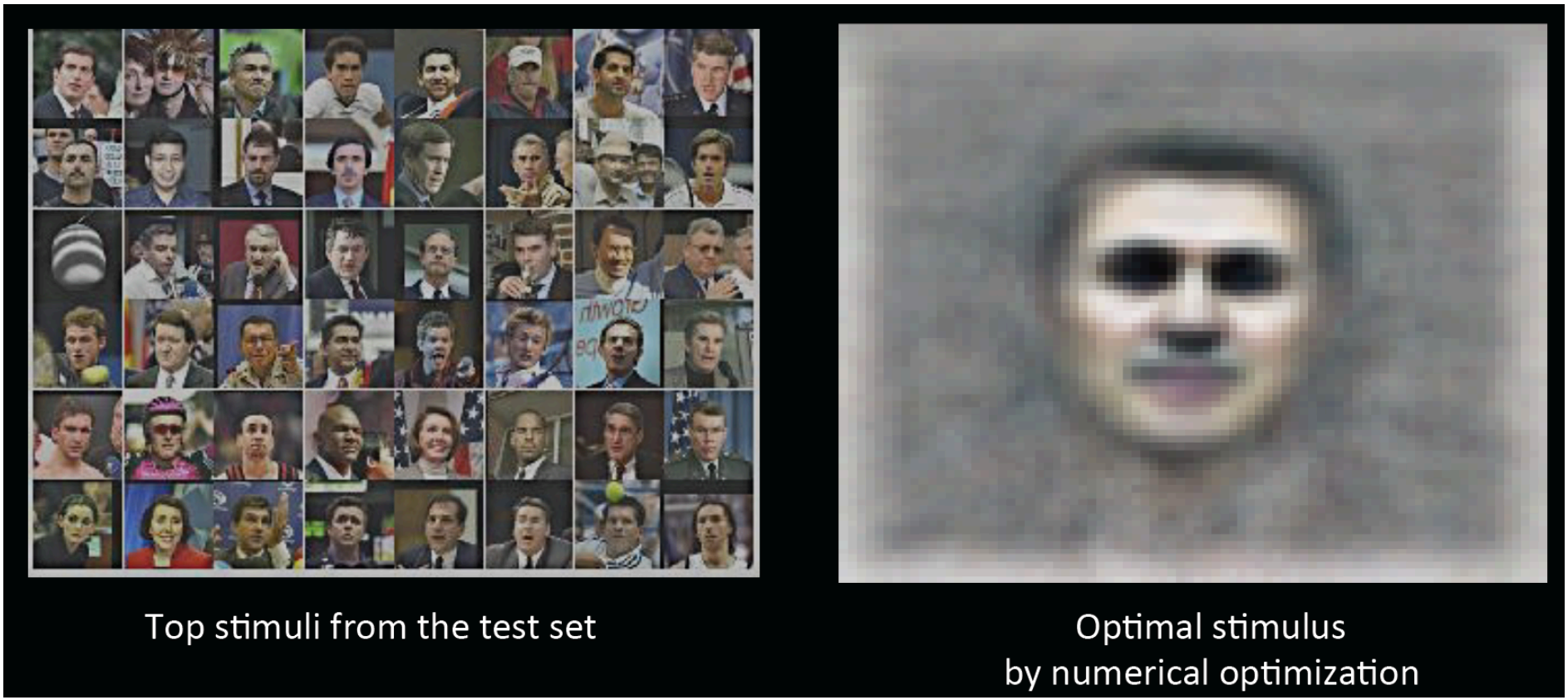
Deep network has 1 billion parameters, trained on 16k cores for a week

Cat neuron



*Graphics from [Le et al., 2012]

Face neuron



*Graphics from [Le et al., 2012]

Deep Learning と他の機械学習手法との関連

- RBM = product-of-expert model (distributed representation of data)
  - Like a mixture model with $2^{|h|}$ hidden components
    $p(x) = \sum_h p(h)p(x|h)$, but much more compact
  - Like a multi-clustering, with less loss of information than clustering
- Lower layers of Neural Net as kernel for SVM [Li et al., 2005]
- Decision trees are deep (but no distributed representation). Random forests are both deep and distributed. (They work well in practice too!)
- Philosophical connections to:
  - Semi-supervised Learning: exploit both labeled and unlabeled data
  - Curriculum Learning: start on easy task, gradually level-up
  - Multi-task Learning: learn and share sub-tasks
- It's important to understand that Deep Learning does not solve everything. It is just one method.

最先端の最先端

- SGD alternative [Martens, 2010]
- Better regularization, e.g. Dropout [Hinton et al., 2012b]
- Scaling to large data [Dean et al., 2012]
- Hyper-parameter search [Bergstra et al., 2011]

目次

最近の Deep/Neural に関連する論文

- Sequence labeling
  - [Wang and Manning, 2013, Collobert et al., 2011, Turian et al., 2010]
- Syntax & Morphology
  - [Socher et al., 2013a, Billingsley and Curran, 2012, Hermann and Blunsom, 2013, Stenetorp, 2013, Luong et al., 2013]
- Semantics
  - [Tsubaki et al., 2013, Hashimoto et al., 2013, Srivastava et al., 2013]
- Machine Translation
  - [Liu et al., 2013, Auli et al., 2013, Zou et al., 2013, Kalchbrenner and Blunsom, 2013]
- Text Classification & Applications
  - [Glorot et al., 2011, Socher et al., 2013b, Socher et al., 2011]

Warning!

- There is no consensus yet on how Deep Learning fits with NLP!
- What we'll do: Summarize 2 ways Deep/Neural ideas are used so far
  1. As non-linear classifier (complex feature combination)
  2. As distributed representation (of words/phrases)
- Show one successful and one unsuccessful case study each
- Purpose is to give a sense of the detail design decisions involved, while emphasizing: This is how things are now, but it is not necessary correct!

非線形分類器としての応用

Idea: Directly replace a linear classifer used in NLP with a deep network.
Expected to work if:

1. Difficult to engineer effective features
2. Linear classifier underfits (e.g. high training error)

Successful case study:

- Domain Adaptation for Large-Scale Sentiment Classification: a Deep Learning Approach by X. Glorot, A. Bordes, and Y. Bengio (ICML2011)

Unsuccessful case study:

- Machine Translation N-best Re-ranking (preliminary results)

Case Study 1: Domain Adaptation for Large-Scale Sentiment Classification [Glorot et al., 2011]

- Domain adaptation problem in text classification [Blitzer et al., 2007]
  - Amazon review text in 4 domains: electronics, books, DVDs, kitchen.
  - Small labeled data: positive/negative review
  - Large unlabeled data: raw review text
  - Goal: Learn classifier(s) that do well in each domain
- Approach:
  1. Pre-training on unlabeled data to obtain ”good features”
  2. Train SVM on target labeled data, using above features
- Hope: Discover combinations of base features (frequent unigram/bigram) that are generalizable across domains

ネットワークのデザイン

- Input x: binary representation of document
  - 5000 most frequent 1-grams, 2-grams
- First layer: Denoising Auto-Encoder with masking noise:
  - Masking noise: randomly set bit to 0
  - Reconstruction criterion is KL Divergence Loss:
    $-\sum_{i=1}^{dim} x_i \log(r_i) + (1 - x_i) \log(1 - r_i), r = \sigma(W^T \sigma(W \cdot \tilde{x}))$
- Higher layers: Denosing Auto-encoder with Gausian noise and rectifier
  - Rectifier non-linearity: $g(\cdot) = max(W \cdot \tilde{x}, 0)$ encourages sparsity
  - Reconstruction criterion is Squared Loss: $||x - \sigma(W^T g(W \cdot \tilde{x}))||^2$
- Output layer: SVM
- Hyperparameters:
  - Masking noise (best=0.8), Gaussian noise stddev, number of hidden layers (1-3), size of hidden layers (1000, 2500, best=5000), L1 regularization tradeoff, learning rate γ

実験結果 [Glorot et al., 2011]

Evaluation metric:

- $error(A, B)$ = error of classifier trained on Domain A, test on Domain B
- transfer loss = $error(Source, Target) - error(Target, Target)$

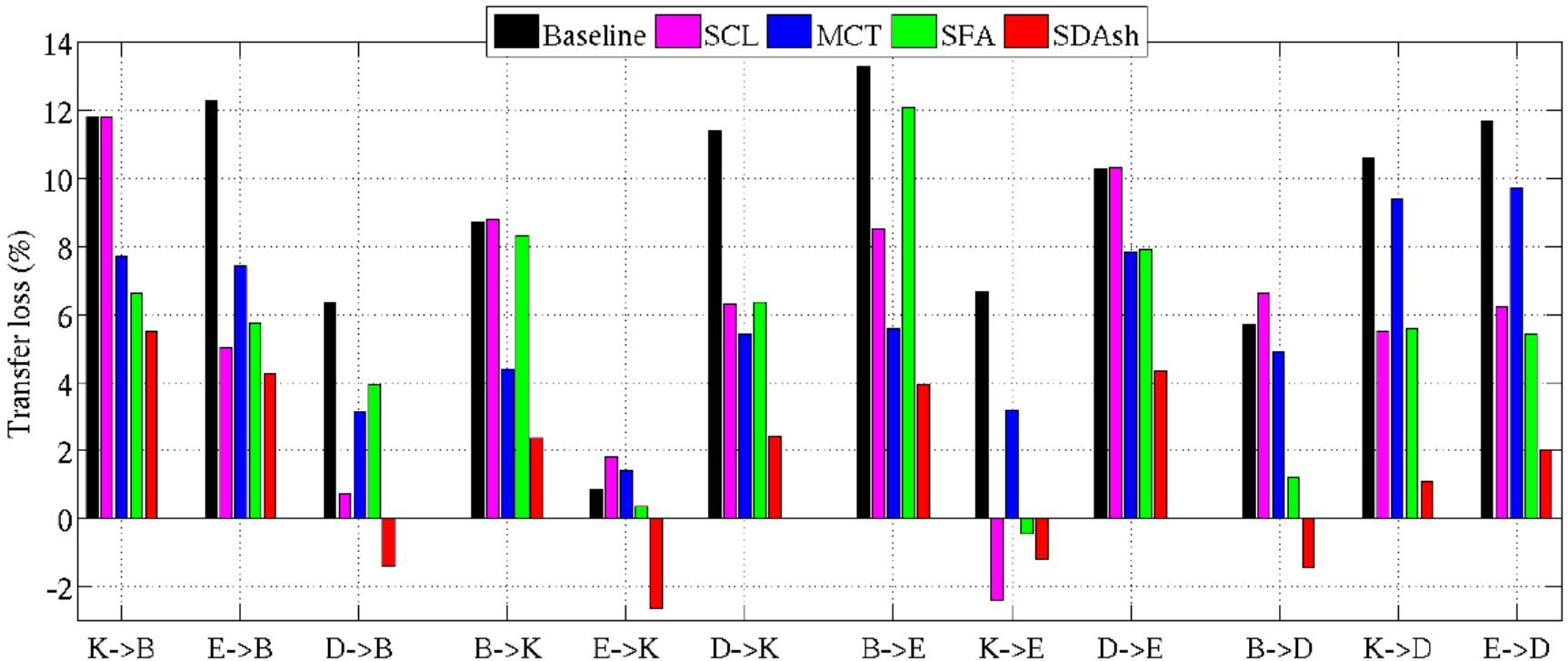
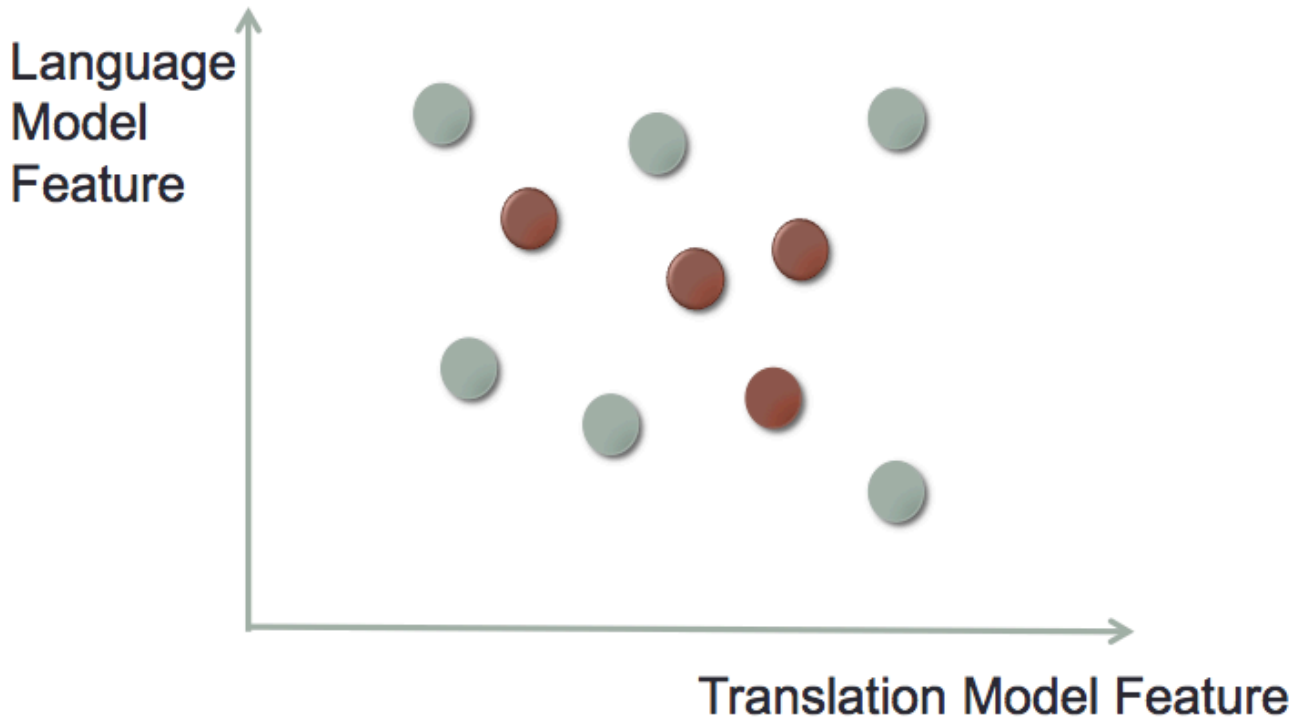


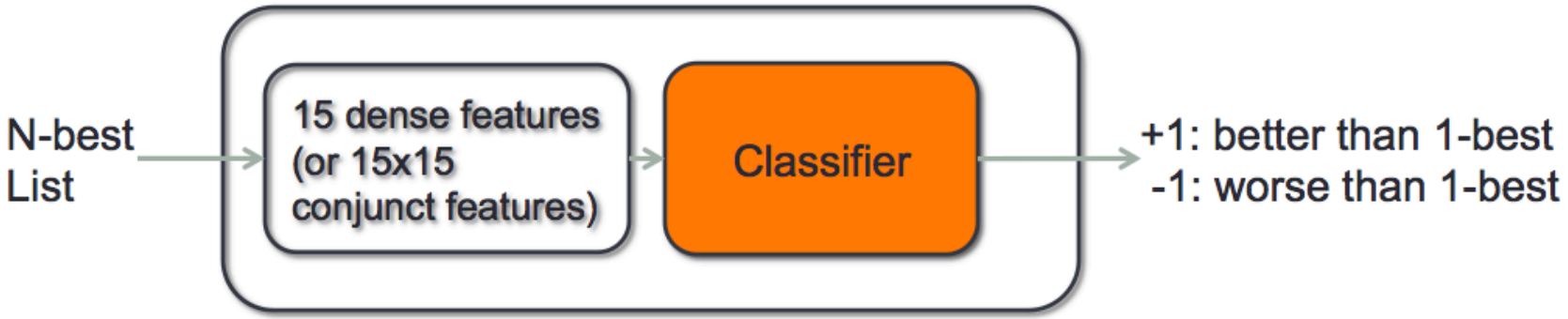
Figure 1. **Transfer losses on the Amazon benchmark** of 4 domains: *Kitchen*(K), *Electronics*(E), *DVDs*(D) and *Books*(B). All methods are trained on the labeled set of one domain and evaluated on the test sets of the others. SDA_{sh} outperforms all others on 11 out of 12 cases.

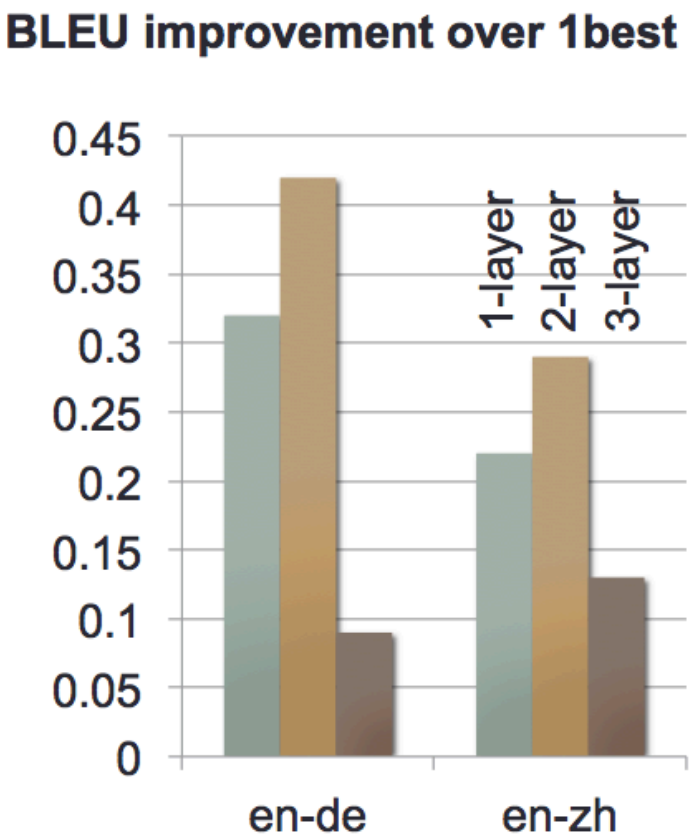
Case Study 2: 機械翻訳 N-best List Re-ranking

- Hypothesis: Dense features in current systems are not sufficiently expressive [Duh and Kirchhoff, 2008]
  - Translation model, language model scores are too coarse-grained
  - Linear re-ranker attains only convex hull of N-best candidates



予備実験





Issue:
3-layer net overfits on wrong objective

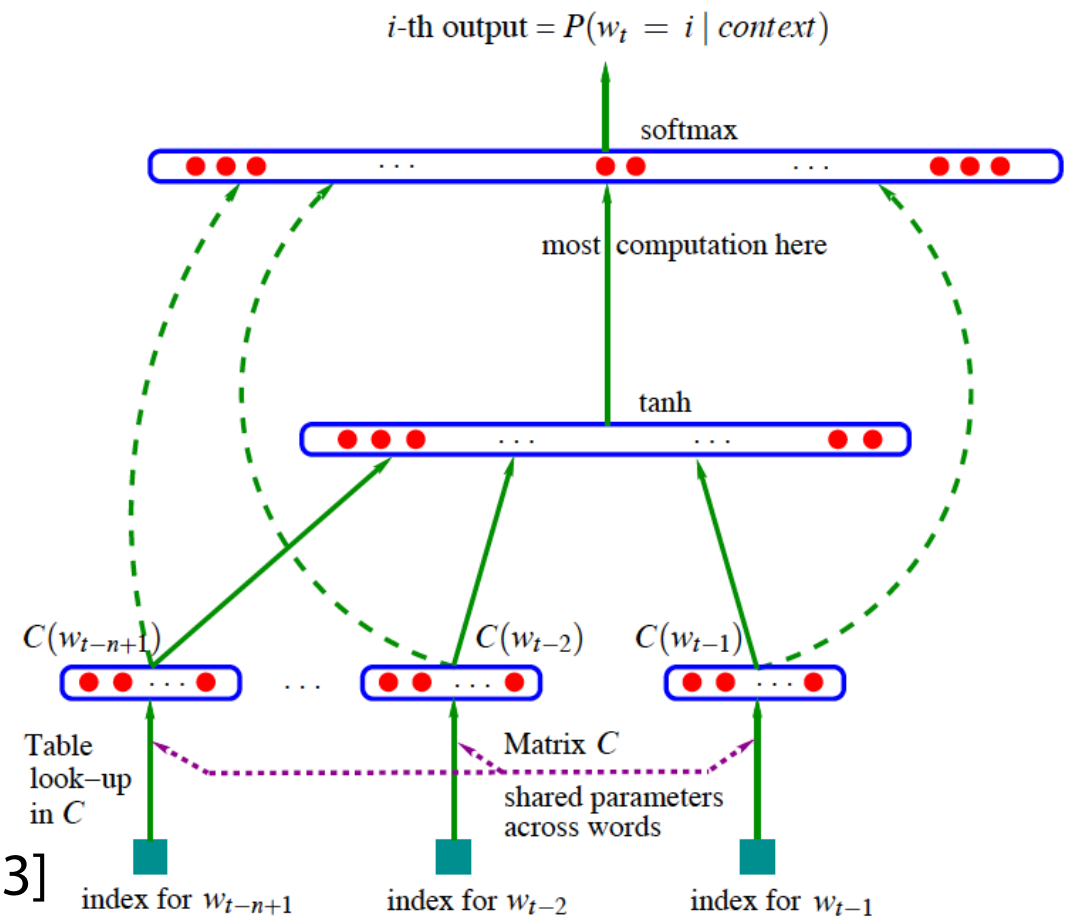
目次

1. Deep Learning の背景
   - ニューラルネットワーク（一層、二層）
   - 深層構造の可能性と限界
   - 2006 年のブレークスルー
2. 主な深層構造 (Deep Architecture)
   - Deep Belief Nets (DBN) [Hinton et al., 2006]
   - Stacked Auto-Encoders (SAE) [Bengio et al., 2006]
   - 機械学習の分野における Deep Learning の現状
3. 言語処理への応用
   - 非線形分類器としての応用 (Non-linear classifier)
   - 分散表現としての応用 (Distributed representation)
   - ディスカッション

単語の分散表現 (Distributed Representation of Words)

- One-hot representation is common in NLP:
  - ″dog″=$[1, 0, 0, \ldots, 0]$, (vector dimension = vocabulary size)
  - ″cat″=$[0, 1, 0, \ldots, 0]$
  - ″the″=$[0, 0, 0, \ldots, 1]$
  - ″dog″ and ″cat″ share zero similarity, just like ″dog″ and ″the″
- Distributed represention models word as vectors:
  - ″dog″=$[1, 0, 0.9, 0.0]$
  - ″cat″=$[1, 0, 0.5, 0.2]$
  - ″the″=$[0, 1, 0.0, 0.0]$
  - Hopefully, captures nuanced factors like POS & semantics

Neural Net による単語の分散表現の学習



[Bengio et al., 2003]

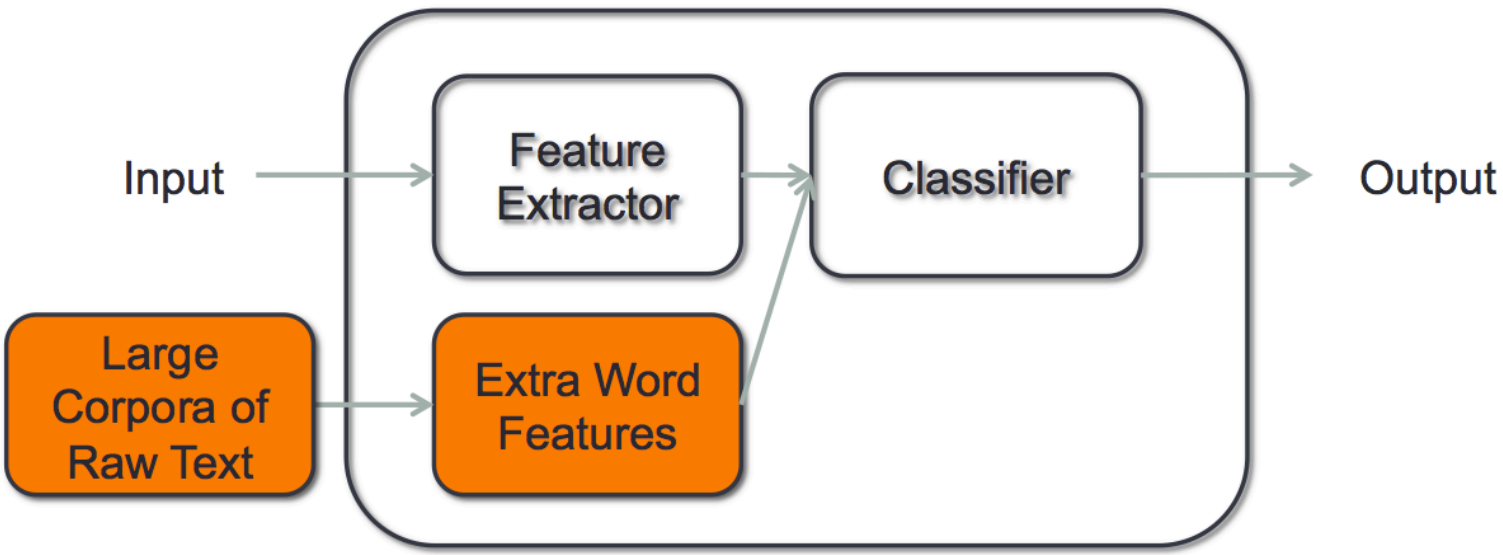
- $C()$ are the distributed representations (embeddings)
- The history context $[C(w_{t-1})\ C(w_{t-2})\ C(w_{t-3})]$ is compressed to a h node hidden layer via $tanh(Hx)$
- Final output mapping with softmax gives probabilities $p(w_t|context)$.

分散表現としての応用

Idea: Replace/Append original features with distributed representation.

Expected to work if:

1. Original features are too sparse (e.g. small training data)
2. Distributed representation enables more flexible model of language



Unsuccessful case study:

- POS Tagging of Web Text (preliminary results)

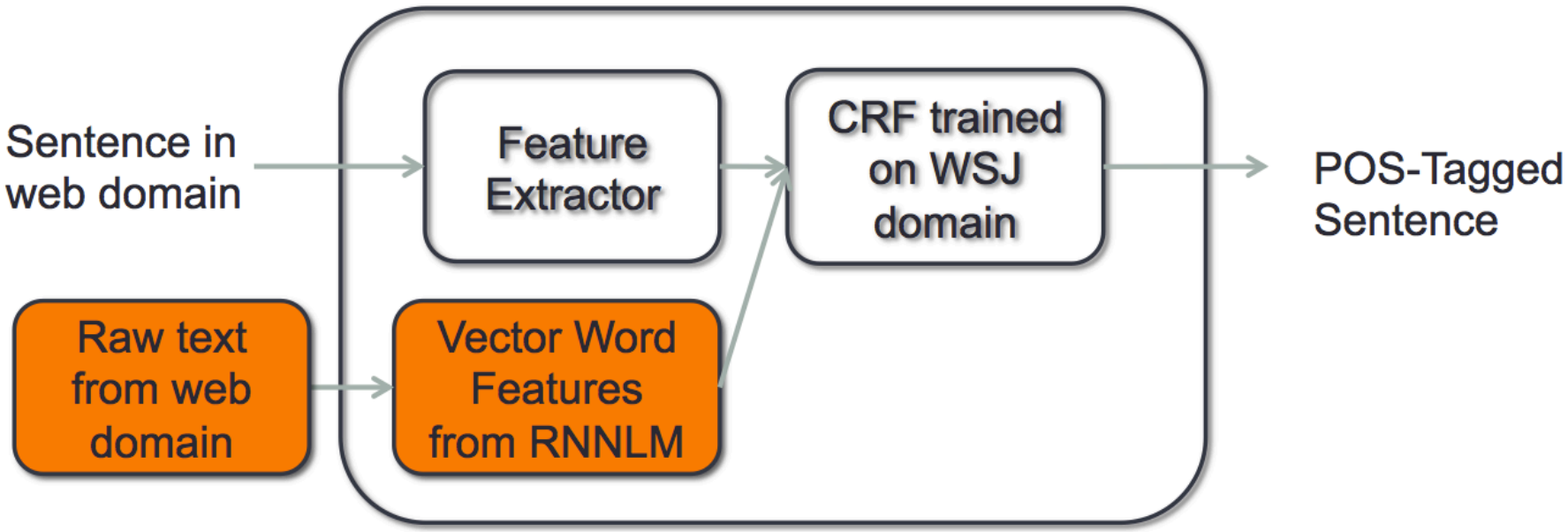
Successful case study:

- Parsing with Compositional Vector Grammers, by Socher+ (ACl2013)

Case Study 1: POS tagging of Web Text

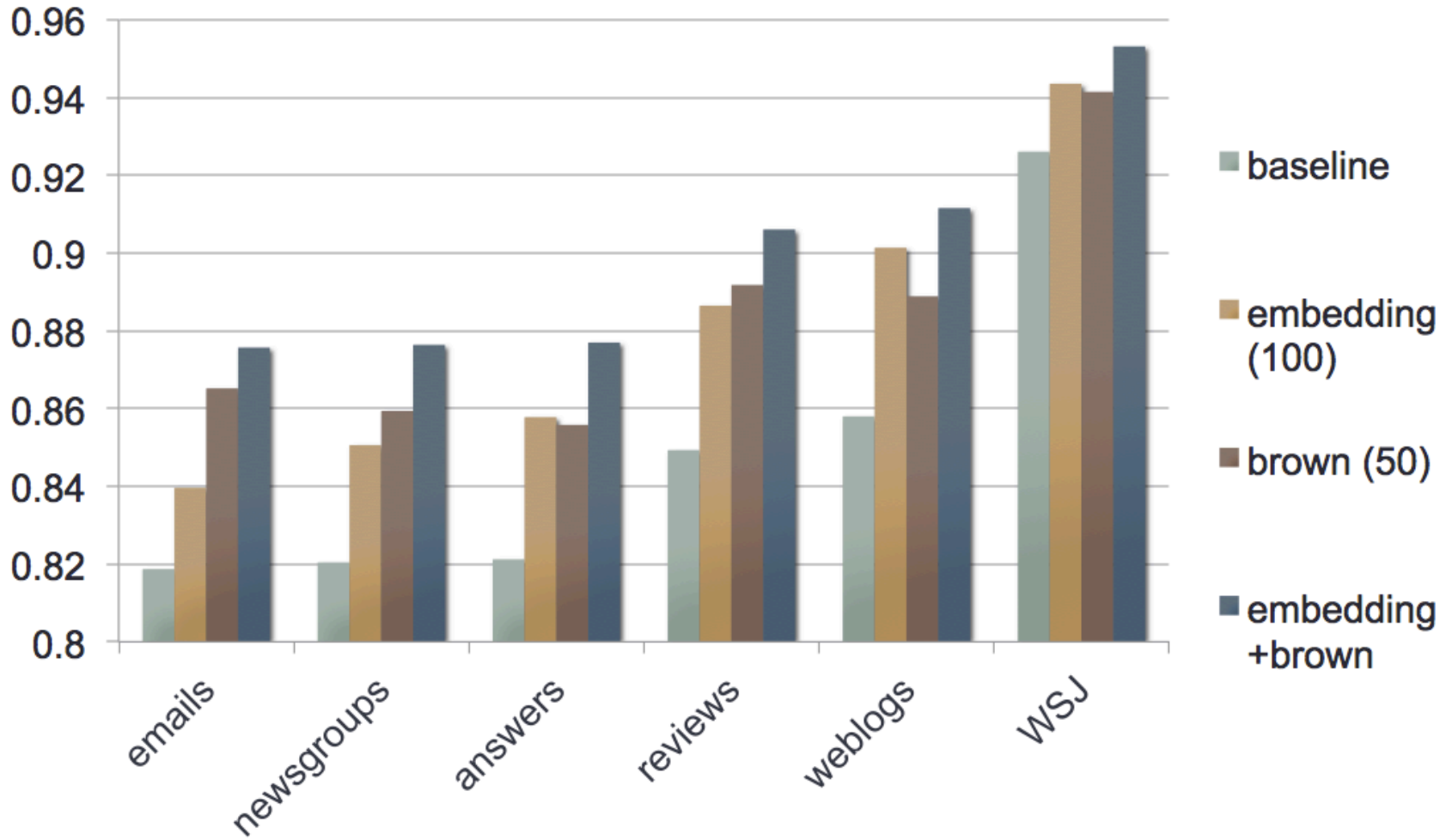
Motivation: POS tagging performs well on WSJ (news) but poorly on web text. One main reason is unknown words.

- Distributed representation ensures unknown words can be tagged because similar words occur in WSJ Treebank



*There are many ways to learn word representations [Chen et al., 2013]. Here, RNNLM = Recurrent Neural Net Language Model [Mikolov et al., 2011]

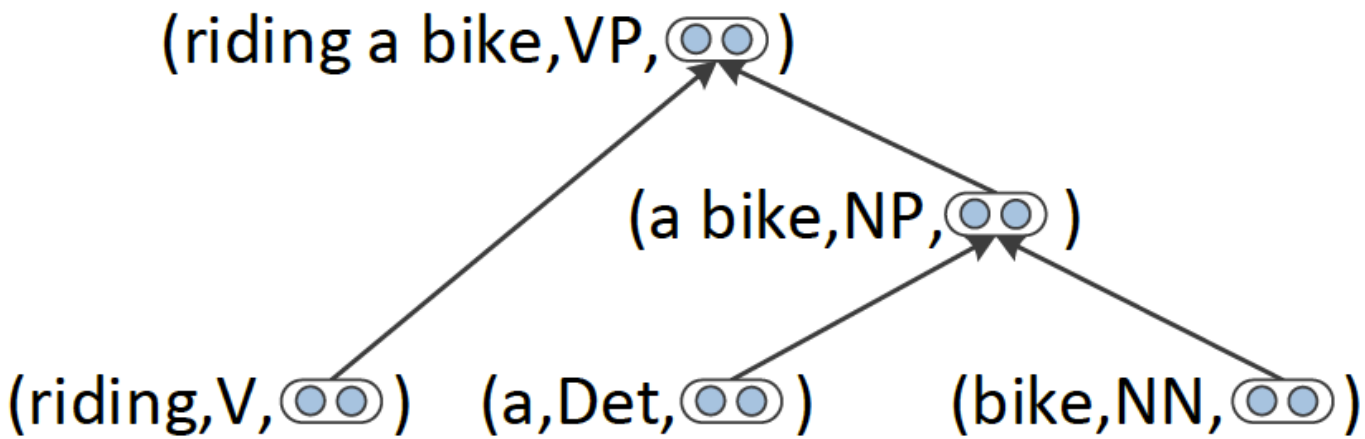
POS Tagging 正解率 (SANCL2012 Shared Task data)



Conclusion:

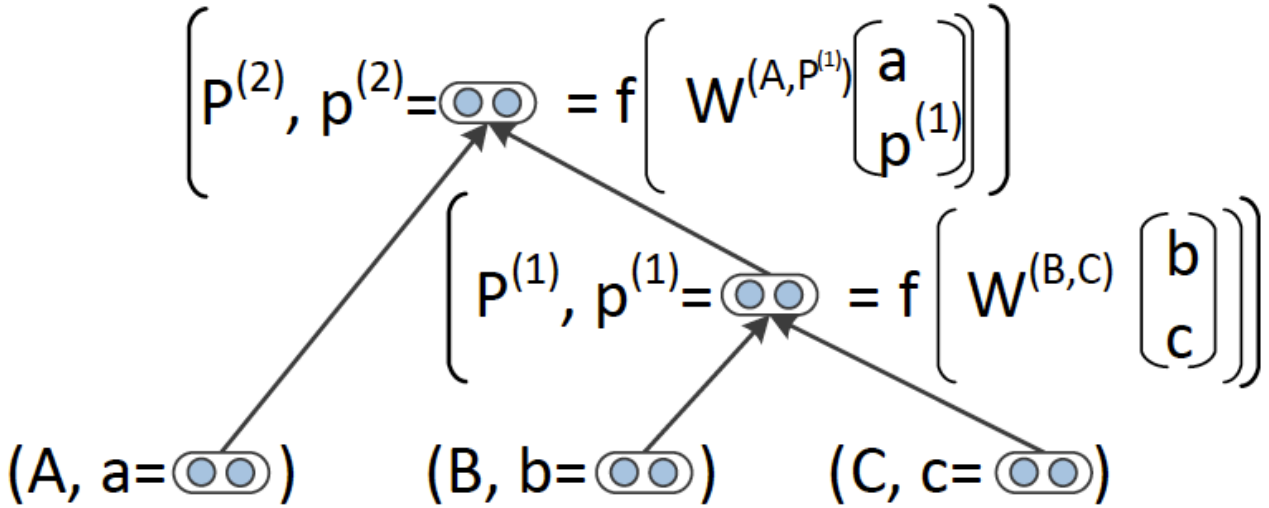
- Distributed representation helps. But Brown clustering is just as good.
- What to improve: Representation? CRF integration? Joint learning?

Case Study 2: Parsing with Compositional Vector Grammar [Socher et al., 2013a]



Parsing results can be improved by splitting coarse categories (e.g. NP, VP).
Rather than splitting, directly learn distributed representation of phrases

Compositional Vector Grammar [Socher et al., 2013a]



- Begin with word representation. Phrase vector is output of 1 layer net, compute recursively bottom-up.
- The score of e.g. node $p^{(1)}$ is $v^{(B,C)}p^{(1)} + \log P(P_1 \rightarrow BC)$
- Predicted parse tree is the one that achieves max sum of node scores.
- Weight matrix (for each node type) is trained by structured max-margin objective

WSJ Section 23 構文解析結果

System	F1
Stanford Parser (PCFG)	85.5
Stanford Parser (Factored)	86.6
Berkeley Parser	90.1
Compositional Vector Grammar	90.4

End-to-end distributed representation outperforms both manually factored and automatically split state systems.

Case Study まとめ

Exploiting Non-linear Classifiers

- It's possible to directly apply Deep Learning to text problems with little modification, as evidenced by [Glorot et al., 2011]
- But sometimes NLP-specific modifications are needed, e.g. training objective mismatch in Machine Translation N-best experiment

Exploiting Distributed Representation

- Distributed Word Representation is a simple way to improve robustness of NLP, but it's not the only way (POS tagging experiment)
- Promising direction: distributed representations beyond words, considering syntax, compositionality, and various language phenomena, e.g. [Socher et al., 2013a]

目次

ディスカッション

- Is Deep Learning just a fad?
- What ideas in Deep Learning will stand the test of time?
- Can Deep Learning really help NLP? Isn't a single word already an extremely informative feature, worth a thousand pixels?

まとめ

1. Deep Learning の背景
   - ニューラルネットワーク（一層、二層）
   - 深層構造の可能性と限界
   - 2006 年のブレークスルー
2. 主な深層構造 (Deep Architecture)
   - Deep Belief Nets (DBN) [Hinton et al., 2006]
   - Stacked Auto-Encoders (SAE) [Bengio et al., 2006]
   - 機械学習の分野における Deep Learning の現状
3. 言語処理への応用
   - 非線形分類器としての応用 (Non-linear classifier)
   - 分散表現としての応用 (Distributed representation)
   - ディスカッション

もっと知りたい方は

- Survey paper:
  - Yoshua Bengio's [Bengio, 2009] short book: Learning Deep Architectures for AI[1]
- Courses & In-depth Lecture Notes:
  - My course @ NAIST: `http://cl.naist.jp/~kevinduh/a/deep2014/`
  - Hugo Larochelle's course @ Sherbrooke[2]
- Tutorials for NLPers:
  - IBIS2013 企画セッション：自然言語処理分野におけるディープラーニングの現状[3] by 渡邉陽太郎　氏
  - Richard Socher et. al.'s NAACL2013 tutorial[4]
- To Learn Even More:
  - Theano code samples[5]
  - Blog at `http://deeplearning.net`

[1] `http://www.iro.umontreal.ca/~bengioy/papers/ftml.pdf`
[2] `http://tinyurl.com/qccl66y`
[3] `http://ibisml.org/archive/ibis2013/pdfs/ibis2013-watanabe.pdf`
[4] `http://www.socher.org/index.php/DeepLearningTutorial/`
[5] `http://deeplearning.net/tutorial/`

Thanks for your attention! Questions?

Acknowledgments:

I'd like to thank the students who work with me on this research!

- Daniel Fried (Representation Learning Algorithms)
- Sorami Hisamoto (Dependency Parsing)
- Xiaodong Liu (Chinese Word Segmentation)
- Masashi Tsubaki (Compositional Semantics)
- Akifumi Yoshimoto (GPGPU Hardware)
- Ippei Yoshimoto (ESL Error Detection)

References:

Auli, M., Galley, M., Quirk, C., and Zweig, G. (2013).
Joint language and translation modeling with recurrent neural networks.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA. Association for Computational Linguistics.

Bengio, Y. (2009).
Learning Deep Architectures for AI, volume Foundations and Trends in Machine Learning.
NOW Publishers.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003).
A neural probabilistic language models.
JMLR.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006).
Greedy layer-wise training of deep networks.
In *NIPS'06*, pages 153–160.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégel, B. (2011).
Algorithms for hyper-parameter optimization.
In *Proc. Neural Information Processing Systems 24 (NIPS2011)*.

Billingsley, R. and Curran, J. (2012).
Improvements to training an RNN parser.
In *Proceedings of COLING 2012*, pages 279–294, Mumbai, India. The COLING 2012 Organizing Committee.

Bishop, C. (1995).
Neural Networks for Pattern Recognition.
Oxford University Press.

Blitzer, J., Dredze, M., and Pereira, F. (2007).
Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification.
In *ACL*.

Bourlard, H. and Kamp, Y. (1988).
Auto-association by multilayer perceptrons and singular value decomposition.
Biological Cybernetics, 59:291–294.

Carreira-Perpinan, M. A. and Hinton, G. E. (2005).
On contrastive divergence learning.
In *AISTATS*.

Chen, Y., Perozzi, B., Al-Rfou, R., and Skiena, S. (2013).
The expressive power of word embeddings.
In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011).
Natural language processing (almost) from scratch.
Journal of Machine Learning Research, 12:2493–2537.

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012).
Large scale distributed deep networks.
In *Neural Information Processing Systems (NIPS)*.

Duh, K. and Kirchhoff, K. (2008).
Beyond log-linear models: Boosted minimum error rate training for n-best re-ranking.
In *Proceedings of ACL-08: HLT, Short Papers*, pages 37–40, Columbus, Ohio. Association for Computational Linguistics.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P., and Bengio, S. (2010).
Why does unsupervised pre-training help deep learning?
Journal of Machine Learning Research, 11:625–660.

Erhan, D., Manzagol, P., Bengio, Y., Bengio, S., and Vincent, P. (2009).
The difficulty of training deep architectures and the effect of unsupervised pre-training.
In *AISTATS*.

Glorot, X., Bordes, A., and Bengio, Y. (2011).
Domain adaptation for large-scale sentiment classication: A deep learning approach.
In *ICML*.

Hashimoto, K., Miwa, M., Tsuruoka, Y., and Chikayama, T. (2013).
Simple customization of recursive neural networks for semantic relation classification.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376, Seattle, Washington, USA. Association for Computational Linguistics.

Hermann, K. M. and Blunsom, P. (2013).
The role of syntax in vector space models of compositional semantics.
In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria. Association for Computational Linguistics.

Hinton, G., Deng, L., Yu, D., Dahl, G., A.Mohamed, Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012a).
Deep neural networks for acoustic modeling in speech recognition.
IEEE Signal Processing Magazine, 29.

Hinton, G., Osindero, S., and Teh, Y.-W. (2006).
A fast learning algorithm for deep belief nets.
Neural Computation, 18:1527–1554.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012b).
Improving neural networks by preventing co-adaptation of feature detectors.
CoRR, abs/1207.0580.

Kalchbrenner, N. and Blunsom, P. (2013).
Recurrent continuous translation models.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., and Ng, A. Y. (2012).
Building high-level features using large scale unsupervised learning.
In *ICML*.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. (2009).
Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.
In *ICML*.

Li, X., Bilmes, J., and Malkin, J. (2005).
Maximum margin learning and adaptation of MLP classifiers.
In *Interspeech*.

Liu, L., Watanabe, T., Sumita, E., and Zhao, T. (2013).
Additive neural networks for statistical machine translation.
In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 791–801, Sofia, Bulgaria. Association for Computational Linguistics.

Luong, T., Socher, R., and Manning, C. (2013).
Better word representations with recursive neural networks for morphology.
In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Martens, J. (2010).
Deep learning via Hessian-free optimization.
In *Proceedings of the 27th International Conference on Machine Learning (ICML)*.

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Černocký, J. (2011).
Strategies for training large scale neural network language model.
In *ASRU*.

Ranzato, M., Boureau, Y.-L., and LeCun, Y. (2006).
Sparse feature learning for deep belief networks.
In *NIPS*.

Salakhutdinov, R. and Hinton, G. (2009).
Deep Boltzmann machines.
In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455.

Socher, R., Bauer, J., Manning, C. D., and Andrew Y., N. (2013a).
Parsing with compositional vector grammars.
In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Socher, R., Huang, E. H., Pennin, J., Ng, A. Y., and Manning, C. D. (2011).
Dynamic pooling and unfolding recursive autoencoders for paraphrase detection.
In *NIPS*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013b).
Recursive deep models for semantic compositionality over a sentiment treebank.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Srivastava, S., Hovy, D., and Hovy, E. (2013).
A walk-based semantically enriched tree kernel over distributed word representations.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1411–1416, Seattle, Washington, USA. Association for Computational Linguistics.

Stenetorp, P. (2013).
Transition-based dependency parsing using recursive neural networks.
In *NIPS 2013 Deep Learning Workhop*.

Tsubaki, M., Duh, K., Shimbo, M., and Matsumoto, Y. (2013).
Modeling and learning semantic co-compositionality through prototype projections and neural networks.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 130–140, Seattle, Washington, USA. Association for Computational Linguistics.

Turian, J., Ratinov, L.-A., and Bengio, Y. (2010).
Word representations: A simple and general method for semi-supervised learning.
In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010).
Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.
Journal of Machine Learning Research, 11:3371–3408.

Vinyals, O., Jia, Y., Deng, L., and Darrell, T. (2012).
Learning with recursive perceptual representations.
In *NIPS*.

Wang, M. and Manning, C. (2013).
Effect of non-linear deep architecture in sequence labeling.
In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*.

Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. (2013).
Bilingual word embeddings for phrase-based machine translation.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA. Association for Computational Linguistics.