

RWTH AACHEN
UNIVERSITY

Advanced Machine Learning Lecture 20

Wrapping Up

04.02.2016

Bastian Leibe
RWTH Aachen
<http://www.vision.rwth-aachen.de/>
leibe@vision.rwth-aachen.de

Advanced Machine Learning Winter'15

RWTH AACHEN
UNIVERSITY

Announcements

- Last lecture next Thursday: Repetition
 - Summary of all topics in the lecture
 - “Big picture” and current research directions
 - Opportunity to ask questions
 - *Please use this opportunity and prepare questions!*
- Exam format
 - Exams will be oral
 - Duration: 30 minutes
 - I will give you 4 questions and expect you to answer 3 of them.
 - Each such question will cover material from -1-2 lecture slots

B. Leibe 2

RWTH AACHEN
UNIVERSITY

This Lecture: Advanced Machine Learning

- Regression Approaches
 - Linear Regression
 - Regularization (Ridge, Lasso)
 - Gaussian Processes
- Learning with Latent Variables
 - Prob. Distributions & Approx. Inference
 - Mixture Models
 - EM and Generalizations
- Deep Learning
 - Linear Discriminants
 - Neural Networks
 - Backpropagation & Optimization
 - CNNs, RNNs, RBMs, etc.

B. Leibe

RWTH AACHEN
UNIVERSITY

Topics of This Lecture

- Recap: Restricted Boltzmann Machines
 - Energy based Models
 - RBMs
 - Deep Belief Networks
- Initialization Revisited
 - Analysis
 - Glorot Initialization
 - Extension to ReLU
- Outlook
 - Reinforcement Learning

B. Leibe 4

RWTH AACHEN
UNIVERSITY

Recap: Energy Based Models (EBM)

- Energy Based Probabilistic Models
 - Define the joint probability over a set of variables \mathbf{x} through an energy function

$$p(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})}$$
 where the normalization factor Z is called the **partition function**

$$Z = \sum_{\mathbf{x}} e^{-E(\mathbf{x})}$$
 - An EBM can be learned by performing (stochastic) gradient descent on the negative log-likelihood of the training data

$$\mathcal{L}(\theta, D) = \frac{1}{N} \sum_{x_n \in D} \log p(x_n)$$
 using the stochastic gradient $-\frac{\partial \log p(x_n)}{\partial \theta}$

B. Leibe 5

RWTH AACHEN
UNIVERSITY

Recap: EBMs with Hidden Units

- Expressing the gradient
 - **Free energy** for a model with hidden variables \mathbf{h}

$$\mathcal{F}(\mathbf{x}) = -\log \sum_{\mathbf{h}} e^{-E(\mathbf{x}, \mathbf{h})}$$
 - Free energy formulation of the joint probability

$$p(\mathbf{x}) = \frac{e^{-\mathcal{F}(\mathbf{x})}}{Z} \quad \text{with} \quad Z = \sum_{\mathbf{x}} e^{-\mathcal{F}(\mathbf{x})}$$
 - The negative log-likelihood gradient then takes the following form, which is difficult to determine analytically

$$-\frac{\partial \log p(\mathbf{x})}{\partial \theta} = \underbrace{\frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta}}_{\text{Positive phase}} - \underbrace{\sum_{\tilde{\mathbf{x}}} p(\tilde{\mathbf{x}}) \frac{\partial \mathcal{F}(\tilde{\mathbf{x}})}{\partial \theta}}_{\text{Negative phase}}$$

B. Leibe 6

RWTH AACHEN UNIVERSITY

Recap: Steps Towards a Solution...

- Monte Carlo approximation
 - Estimate the expectation using a fixed number of model samples for the negative phase gradient (“negative particles”)

$$\frac{\partial \log p(\mathbf{x})}{\partial \theta} \approx \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta} - \frac{1}{|\mathcal{N}|} \sum_{\mathbf{x} \in \mathcal{N}} \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta}$$

free energy at current point
avg. free energy for all other points

- With this, we almost have a practical stochastic algorithm for learning an EBM.
- We just need to define how to extract the negative particles \mathcal{N} .
 - Many sampling approaches can be used here.
 - MCMC methods are especially well-suited.

And this is where all parts of the lecture finally come together...

7

RWTH AACHEN UNIVERSITY

Recap: Restricted Boltzmann Machines

- Properties
 - Energy Function of an RBM

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} w_{ij} v_i h_j$$
 - This translates to a free energy formula

$$\mathcal{F}(\mathbf{v}) = -\mathbf{b}^T \mathbf{v} - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i \cdot \mathbf{v})}$$
 - Factorization property

$$p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v})$$

$$p(\mathbf{v}|\mathbf{h}) = \prod_j p(v_j|\mathbf{h})$$
 - RBMs can be seen as a product of experts specializing on different areas and detecting negative constraints.

8

RWTH AACHEN UNIVERSITY

Recap: RBMs with Binary Units

- Binary units
 - Free energy

$$\mathcal{F}(\mathbf{v}) = -\mathbf{b}^T \mathbf{v} - \sum_i \log \left(1 + e^{(c_i + W_i \cdot \mathbf{v})} \right)$$
 - This results in the iterative update equations for the gradient log-likelihoods

$$\frac{\partial \log p(\mathbf{v})}{\partial W_{ij}} = \mathbb{E}_{\mathbf{v}} [p(h_i|\mathbf{v}) \cdot v_j] - v_j^{(t)} \cdot \sigma(W_i \cdot \mathbf{v}^{(t)} + c_i)$$

$$\frac{\partial \log p(\mathbf{v})}{\partial c_i} = \mathbb{E}_{\mathbf{v}} [p(h_i|\mathbf{v})] - \text{sigm}(W_i \cdot \mathbf{v}^{(t)})$$

$$\frac{\partial \log p(\mathbf{v})}{\partial b_j} = \mathbb{E}_{\mathbf{v}} [p(v_j|\mathbf{h})] - v_j^{(t)}$$

9

RWTH AACHEN UNIVERSITY

Recap: RBM Learning (Slow)

- Iterative approach
- Start with a training vector on the visible units. Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.
- This implements a Markov chain that we use to approximate the gradient

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} \approx \langle v_i, h_j \rangle^0 - \langle v_i, h_j \rangle^\infty$$
- Better method in practice: Contrastive Divergence

10

RWTH AACHEN UNIVERSITY

Recap: Contrastive Divergence (Fast)

$$\Delta w_{ij} = \varepsilon (\langle v_i, h_j \rangle^0 - \langle v_i, h_j \rangle^1)$$

- A surprising shortcut
 - Start with a training vector on the visible units.
 - Update all the hidden units in parallel.
 - Update the all visible units in parallel to get a “reconstruction”.
 - Update the hidden units again (no further iterations).
 - This does not follow the gradient of the log likelihood. But it works well [Hinton].

11

RWTH AACHEN UNIVERSITY

Historical Perspective

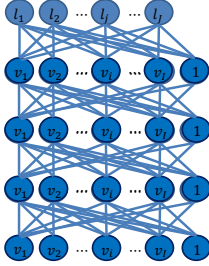
- Training deep networks is difficult
 - Major difficulty: getting the gradient to propagate to the lower layers, so that the weights there can be learned
 - Initialization of the weights plays a major role
 - Weights too small \Rightarrow Signal shrinks from layer to layer
 - Weights too large \Rightarrow Signal grows until it is too massive
 - \Rightarrow Vanishing and exploding gradient problems known from RNNs
- How can we arrive at a good initialization?

12

Advanced Machine Learning Winter'15

Deep Belief Networks (DBN)

- DBN as stacked RBMs
 - RBM: $p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{Z}$
 - $E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h} - \mathbf{v}^T W \mathbf{h}$
 - $p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v})$
 - $p(h_i = 1|\mathbf{v}) = \sigma(\mathbf{c}_i + \mathbf{v}^T W_i)$
- Pre-train each layer from bottom up by considering each pair of layers as an RBM.
- Jointly fine-tune all layers using back-propagation algorithm
- ⇒ Layer-by-layer unsupervised training



Slide credit: Li Deng

B. Leibe

13

Advanced Machine Learning Winter'15

Topics of This Lecture

- Recap: Restricted Boltzmann Machines
 - Energy based Models
 - RBMs
 - Deep Belief Networks
- Initialization Revisited
 - Analysis
 - Glorot Initialization
 - Extension to ReLU
- Outlook
 - Reinforcement Learning

B. Leibe

14

Advanced Machine Learning Winter'15

Glorot Initialization

- Breakthrough results
 - In 2010, Xavier Glorot published an analysis of what went wrong in the initialization and derived a method for automatic initialization.
 - This new initialization massively improved results and made direct learning of deep networks possible overnight.
 - Let's look at his analysis in more detail...

X. Glorot, Y. Bengio, [Understanding the Difficulty of Training Deep Feedforward Neural Networks](#), AISTATS 2010.

B. Leibe

15

Advanced Machine Learning Winter'15

Effect of Sigmoid Nonlinearities

- Effects of sigmoid/tanh function
 - Linear behavior around 0
 - Saturation for large inputs
- If all parameters are too small
 - Variance of activations will drop in each layer
 - Sigmoids are approximately linear close to 0
 - Good for passing gradients through, but...
 - Gradual loss of the nonlinearity
 - ⇒ No benefit of having multiple layers
- If activations become larger and larger
 - They will saturate and gradient will become zero

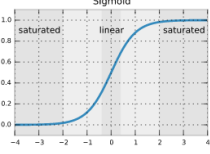


Image source: <http://deendish.in/2015/02/24/network-initialization/>

B. Leibe

16

Advanced Machine Learning Winter'15

Analysis

- Variance of neuron activations
 - Suppose we have an input X with n components and a linear neuron with random weights W that spits out a number Y .
 - What is the variance of Y ?
 - $$Y = W_1 X_1 + W_2 X_2 + \dots + W_n X_n$$
 - If inputs and outputs have both mean 0, the variance is
 - $$\text{Var}(W_i X_i) = E[X_i]^2 \text{Var}(W_i) + E[W_i]^2 \text{Var}(X_i) + \text{Var}(W_i) \text{Var}(X_i)$$

$$= \text{Var}(W_i) \text{Var}(X_i)$$
 - If the X_i and W_i are all i.i.d, then
 - $$\text{Var}(Y) = \text{Var}(W_1 X_1 + W_2 X_2 + \dots + W_n X_n) = n \text{Var}(W_i) \text{Var}(X_i)$$
 - ⇒ The variance of the output is the variance of the input, but scaled by $n \text{Var}(W_i)$.

B. Leibe

17

Advanced Machine Learning Winter'15

Analysis (cont'd)

- Variance of neuron activations
 - if we want the variance of the input and output of a unit to be the same, then $n \text{Var}(W_i)$ should be 1. This means
 - $$\text{Var}(W_i) = \frac{1}{n} = \frac{1}{n_{\text{in}}}$$
 - If we do the same for the backpropagated gradient, we get
 - $$\text{Var}(W_i) = \frac{1}{n_{\text{out}}}$$
 - As a compromise, Glorot & Bengio propose to use
 - $$\text{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$
 - ⇒ Randomly sample the weights with this variance. That's it.

B. Leibe

18

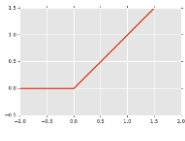
Extension to ReLU

- Another improvement for learning deep models
 - Use Rectified Linear Units (ReLU)

$$g(a) = \max\{0, a\}$$
 - Effect: gradient is propagated with a constant factor

$$\frac{\partial g(a)}{\partial a} = \begin{cases} 1, & a > 0 \\ 0, & \text{else} \end{cases}$$
- We can also improve them with proper initialization
 - However, the Glorot derivation was based on tanh units, linearity assumption around zero does not hold for ReLU.
 - He et al. made the derivations, proposed to use instead

$$\text{Var}(W) = \frac{2}{n_{in}}$$



Advanced Machine Learning Winter'15
B. Leibe
19

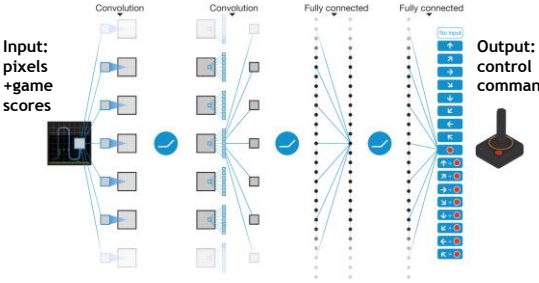
Topics of This Lecture

- Recap: Restricted Boltzmann Machines
 - Energy based Models
 - RBMs
 - Deep Belief Networks
- Initialization Revisited
 - Analysis
 - Glorot Initialization
 - Extension to ReLU
- Outlook
 - Reinforcement Learning

Advanced Machine Learning Winter'15
B. Leibe
20

Outlook: Reinforcement Learning

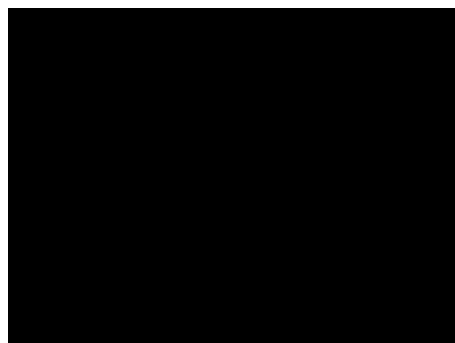
- Learning to play computer games



V. Mnih et al., *Human-level control through deep reinforcement learning*, Nature Vol. 518, pp. 529-533, 2015

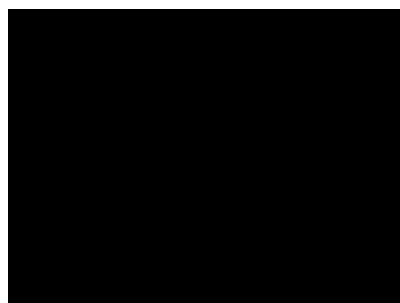
Advanced Machine Learning Winter'15
B. Leibe
21

Results: Space Invaders



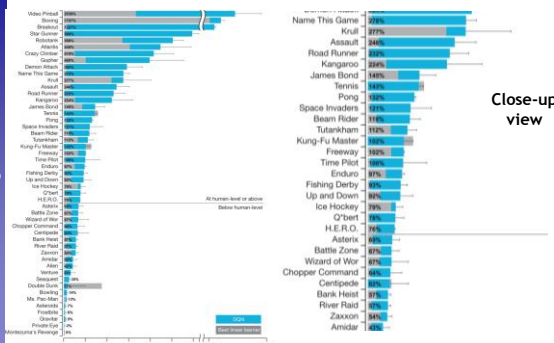
Advanced Machine Learning Winter'15
B. Leibe
22

Results: Breakout



Advanced Machine Learning Winter'15
B. Leibe
23

Comparison with Human Performance



Game	AI Performance (%)
Name This Game	27%
Kufl	27%
Assault	24%
Road Runner	23%
Kangaroo	24%
James Bond	14%
Tennis	14%
Pong	14%
Space Invaders	14%
Beam Rider	11%
Tutankham	11%
Kung Fu Master	10%
Freeway	10%
Time Pilot	9%
Enduro	9%
Fishing Derby	6%
Up and Down	6%
Ice Hockey	7%
O'bert	7%
H.E.R.O.	7%
Battle Zone	7%
Wizard of Wor	6%
Chopper Command	6%
Ceriseade	6%
Bank Heist	6%
River Raid	6%
Zaxxon	6%
Amidar	6%

Advanced Machine Learning Winter'15
B. Leibe
24

Advanced Machine Learning Winter'15

RWTH AACHEN UNIVERSITY

Learned Representation

• t-SNE embedding of DQN last hidden layer (Space Inv.)

25

B. Leibe

Advanced Machine Learning Winter'15

RWTH AACHEN UNIVERSITY

Idea Behind the Model

action values $Q(s,a)$

↑

ConvNet

↑

- Interpretation
 - Assume finite number of actions
 - Each number here is a real-valued quantity that represents the "Q function" in Reinforcement Learning
- Collect experience dataset:
 - Set of tuples $\{(s,a,s',r), \dots\}$
 - (State, Action taken, New state, Reward received)
- L2 Regression Loss

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(\underbrace{r + \gamma \max_{a'} Q(s',a'; \theta_i^-)}_{\text{target value}} - \underbrace{Q(s,a; \theta_i)}_{\text{predicted value}} \right)^2 \right]$$

Current reward + estimate of future reward, discounted by γ

26

Slide credit: Andrei Karapov

B. Leibe

Advanced Machine Learning Winter'15

RWTH AACHEN UNIVERSITY

References and Further Reading

- Initialization
 - X. Glorot, Y. Bengio, [Understanding the difficulty of training deep feedforward neural networks](#), AISTATS 2010.
 - K. He, X. Zhang, S. Ren, J. Sun, [Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification](#), arXiv 1502.01852, 2015.
- ReLu
 - X. Glorot, A. Bordes, Y. Bengio, [Deep sparse rectifier neural networks](#), AISTATS 2011.

31

B. Leibe