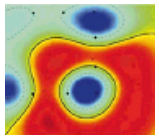
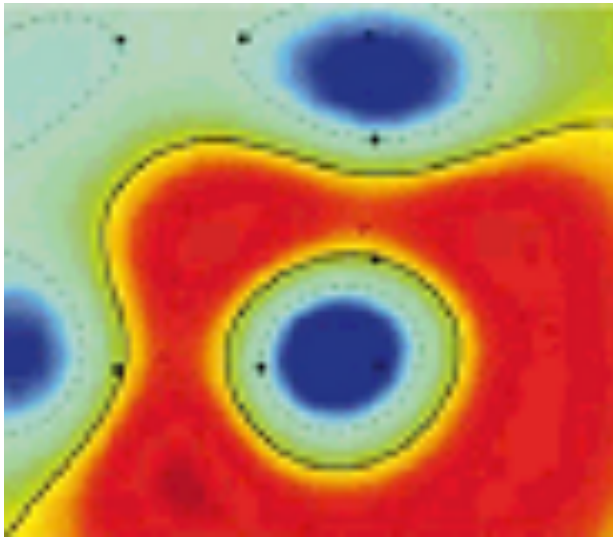


SoSe 2024

Machine Learning 2/2-X

Technische Universität Berlin

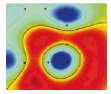


Lecture 13

Neural Networks for Structured Outputs

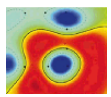
Outline

- Practical Motivations
  - Limitation of standard neural networks
- Simple Structured Prediction Models
  - Sequence-to-Sequence Models
  - Mixture Density Networks
  - Conditional Restricted Boltzmann Machines
- General Framework: Energy-Based Learning
  - Training Procedure and Loss Functions
  - Obtaining Contrastive Examples



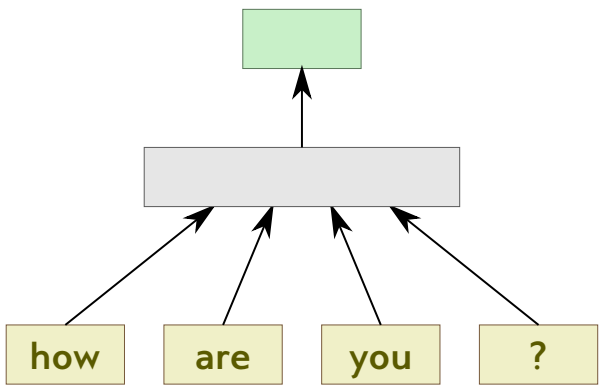
Learning with Structured Output

- Neural networks are generally trained to predict simple quantities such as real values, or class membership.
- In some cases, e.g. machine translation and question-answering systems, it is necessary to predict more complex structures, e.g. a sequence.
- These structured objects are subject to constraints (e.g. local consistency in the sequence, or guided by some underlying dynamics).
- Many methods have been proposed for structured output in the context of neural networks (e.g. sequence-to-sequence models, mixture density networks, energy-based learning).

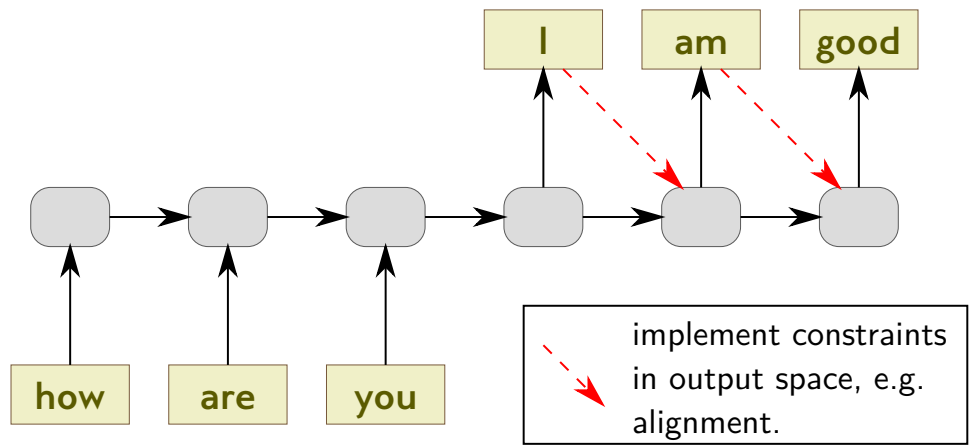


Sequence-to-Sequence Models

Standard neural network

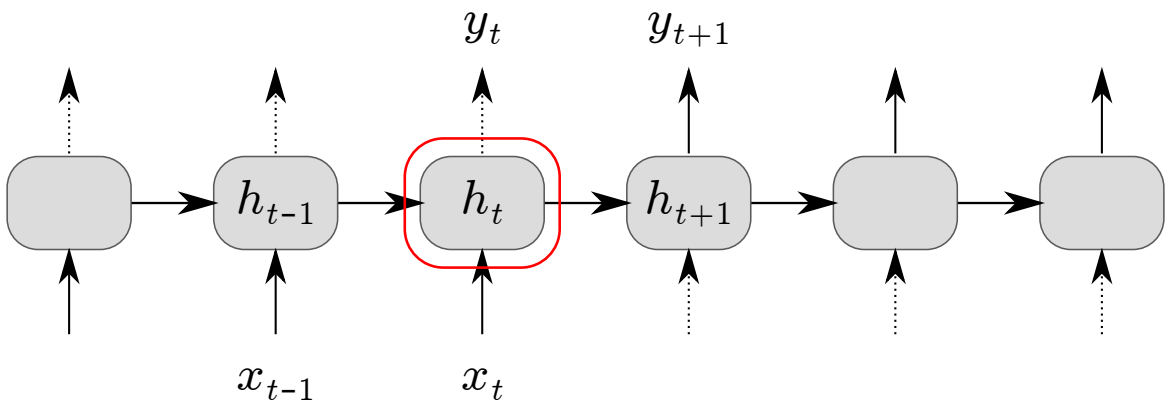


Sequence-to-sequence learning



- In sequence-to-sequence models, e.g. Sutskever et al. [7], the process of generating the output from the input is modeled as a recurrent neural network.
- Generated outputs can be fed back to the neural network (red arrows) to inform on what was generated at the previous step.

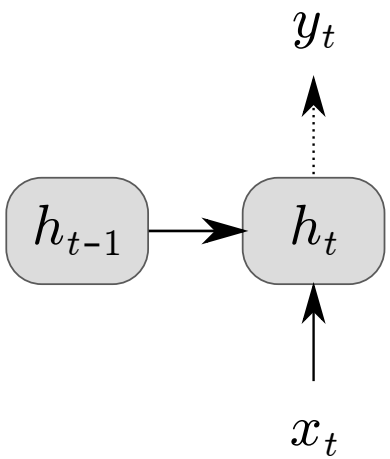
The Recurrent Structure



- Hidden neurons are organized in a sequence. Neurons receive as input the neurons activations from the previous step of the sequence.
- Function f mapping neuron activations between two consecutive time steps is the same at every time step, i.e.

$$h_t = f(h_{t-1}, x_t)$$
$$h_{t+1} = f(h_t, x_{t+1})$$
$$\vdots$$

Vanilla Recurrent Neural Network (RNN)

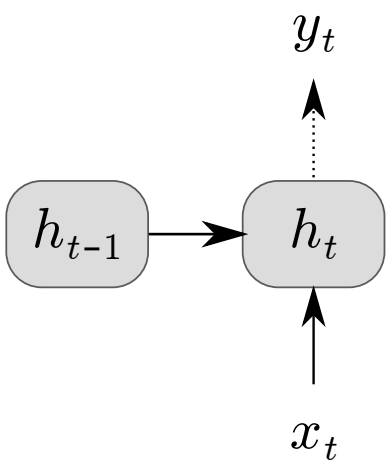


- Two consecutive hidden states $h_{t-1}, h_t \in \mathbb{R}^H$ are related via the equation:

$$h_t = g(Wh_{t-1} + Ux_t)$$

  where g is a nonlinear activation function applied element-wise, e.g. tanh or ReLU.
- At every time step, the state of the system is represented by some H-dimensional real-valued vector. This allows for much more compact state encodings than e.g. hidden Markov models (many states can now fit in fewer dimensions).
- **Problem:** A recurrent neural network is hard to train when the number of time steps is large. (A small variation in the parameters W and U can cause a very large change of representation at the end of the sequence $\rightarrow$ gradient tends to vanish/explode.)

Mitigating the Vanishing-Exploding Gradients



- **Idea:** Replace the vanilla model $h_t = g(Wh_{t-1} + Ux_t)$ with a model that has skip connections:

$$h_t = \underbrace{h_{t-1}}_{\text{skip connection}} + g(Wh_{t-1} + Ux_t)$$

  Parameters W and U only need to encode differences between consecutive time steps (setting them to zero produces a constant time series).

- This enhanced model has an interpretation as the Euler's discretization of the differential equation:

$$\dot{h} = g(Wh + Ux)$$

- More sophisticated approaches work better in practice, e.g. long-short term memories (LSTMs).

Long Short Term Memories (LSTMs)

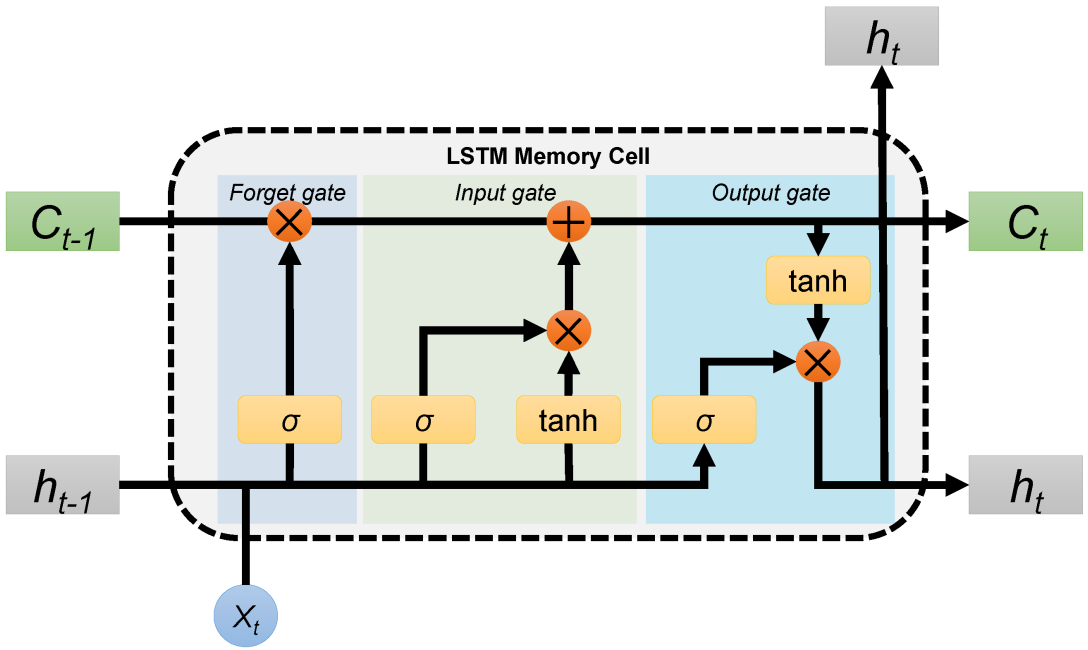
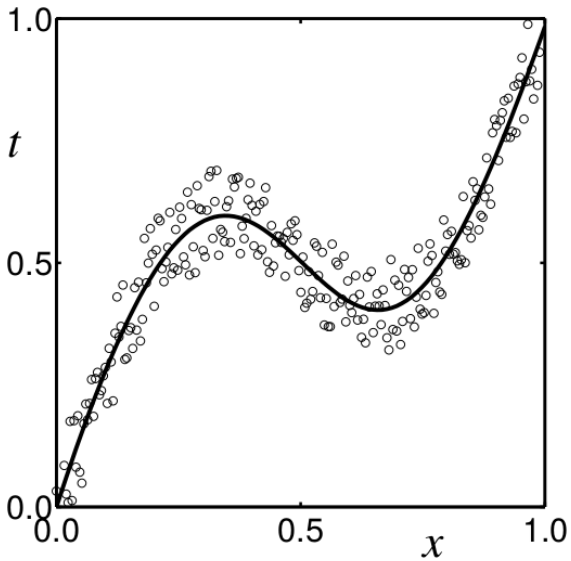


Figure from Fan et al. (2020). Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation

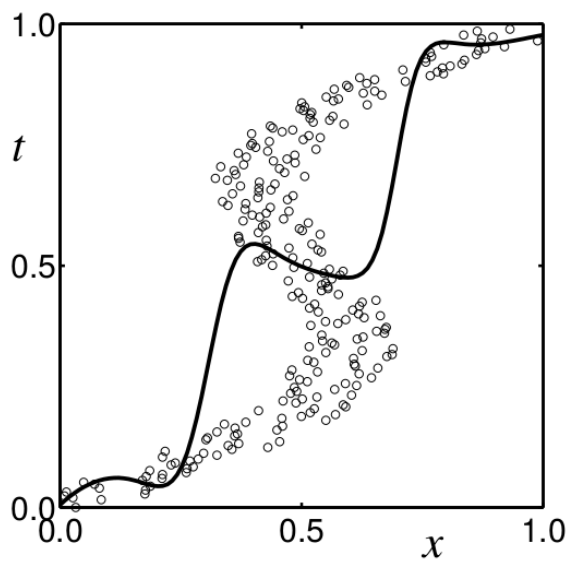
- Core dynamics ($c_{t-1}, c_t, \dots$) are only accessed via gating functions (products $\otimes$ by sigmoids σ). These gating functions can learn what part of the input signal should be kept or filtered out.
- Cell output h_t can be used for prediction, and it can also be reinjected to the cell representing the next time step to inform on what was predicted in the previous time step.

Forward vs. Inverse Problems

Consider now another aspect of structured output which is how to deal with cases where two or more distinct predictions are equally plausible. This often occurs when learning inverse problems.

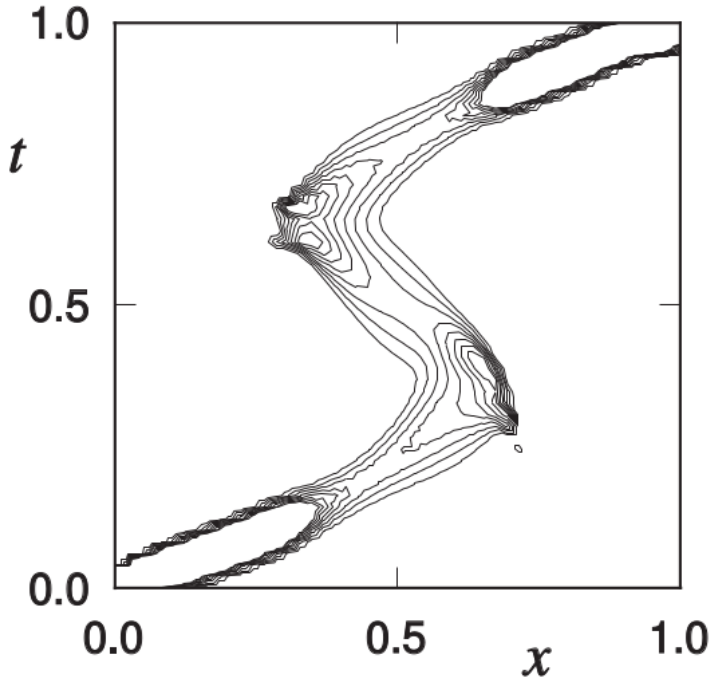


- Structure of the target labels $t = f(x) + \text{noise}$.
- Standard neural networks can be used.



- Problem: f is not invertible.
- For a given input, we need to produce several outputs (or the most likely output).

Learning Output Models



- View the task of prediction as that of learning a conditional probability model $p_\theta(t \mid x)$.
- Minimize the objective

$$\min_\theta \; D(p(t|x) \,\|\, p_\theta(t|x))$$

where D is some divergence measure between two distributions (e.g. Kullback-Leibler Divergence or Wasserstein distance)

Mixture Density Networks (MDNs) [2]

1. Model output as a Gaussian mixture

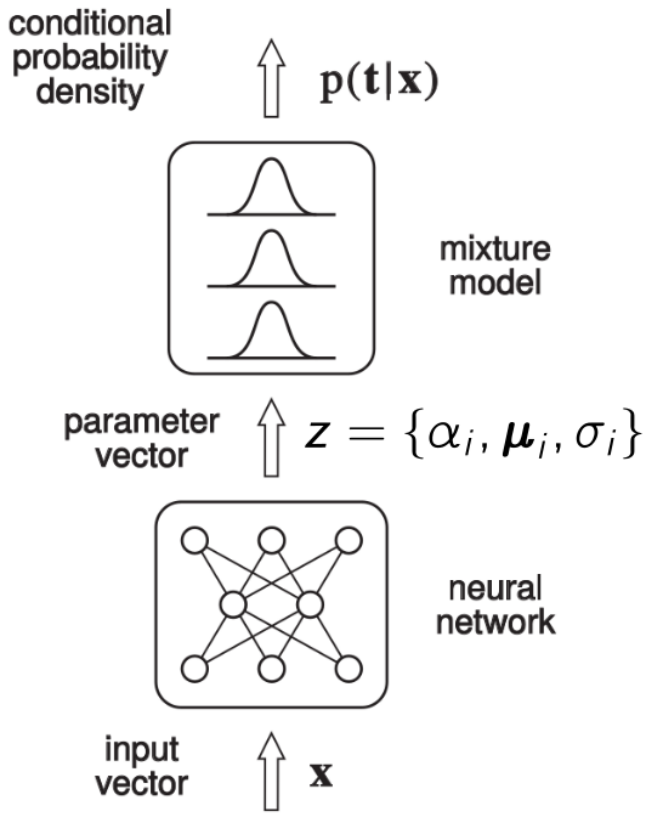
$$p(\mathbf{t} \mid \mathbf{x}) = \sum_{i=1}^{m} \alpha_i(\mathbf{x}) \phi_i(\mathbf{t} \mid \mathbf{x})$$

$$\phi_i(\mathbf{t} \mid \mathbf{x}) = \frac{1}{(2\pi)^{c/2} \sigma_i(\mathbf{x})^c} \exp\left\{ -\frac{\|\mathbf{t} - \boldsymbol{\mu}_i(\mathbf{x})\|^2}{2\sigma_i(\mathbf{x})^2} \right\}$$

where parameters of the mixture are the output of the neural network.

2. Optimize Gaussian mixture's likelihood

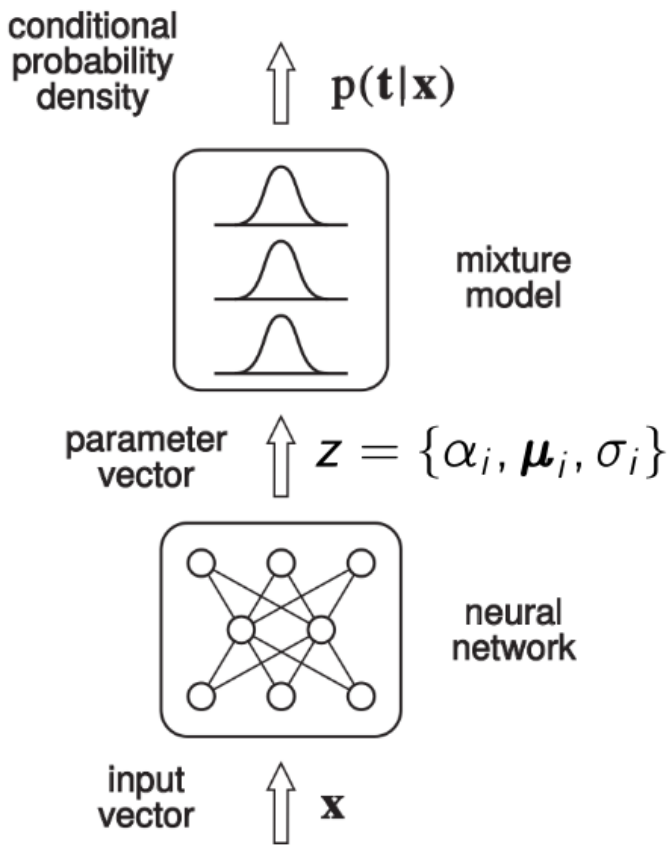
$$E^q = -\ln\left\{ \sum_{i=1}^{m} \alpha_i(\mathbf{x}^q) \phi_i(\mathbf{t}^q \mid \mathbf{x}^q) \right\}$$



Mixture Density Networks (MDNs) [2]

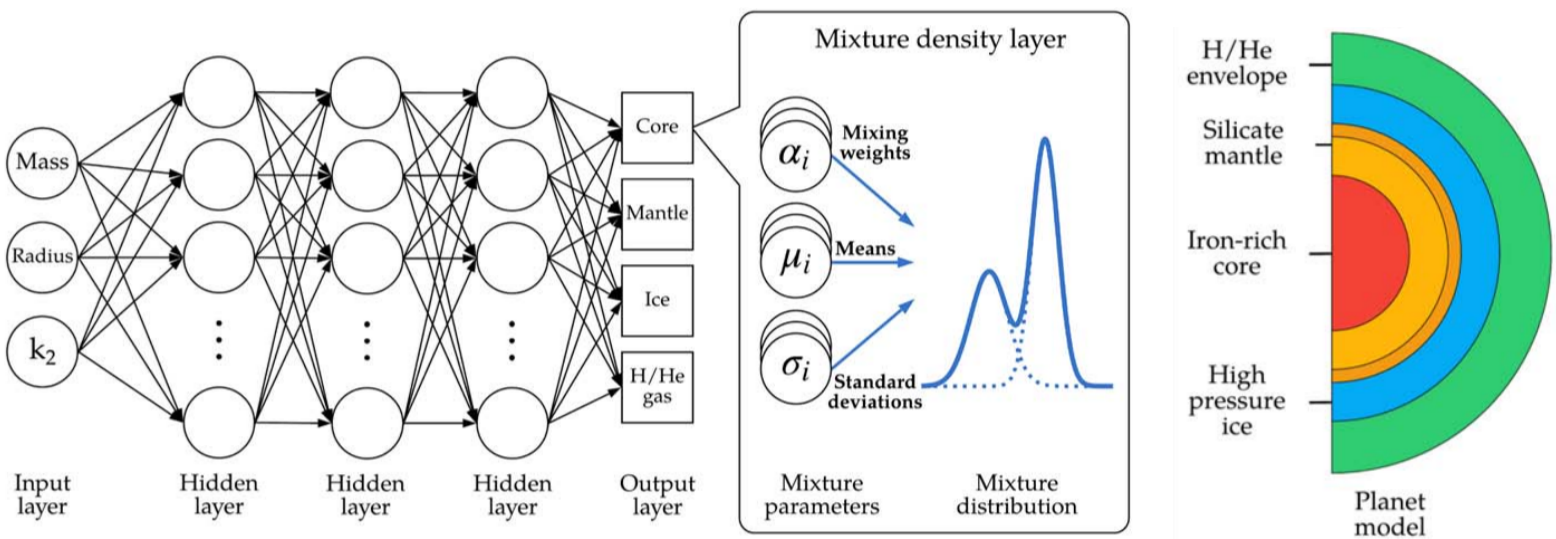
Parameters are subject to constraints.
These constraints can be enforced with an appropriate choice of output layer.

$$\alpha_i = \frac{\exp(z_i^\alpha)}{\sum_{j=1}^{M} \exp(z_j^\alpha)} \qquad \sigma_i = \exp(z_i^\sigma) \qquad \mu_{ik} = z_{ik}^\mu$$



MDNs for Inferring Planets Composition

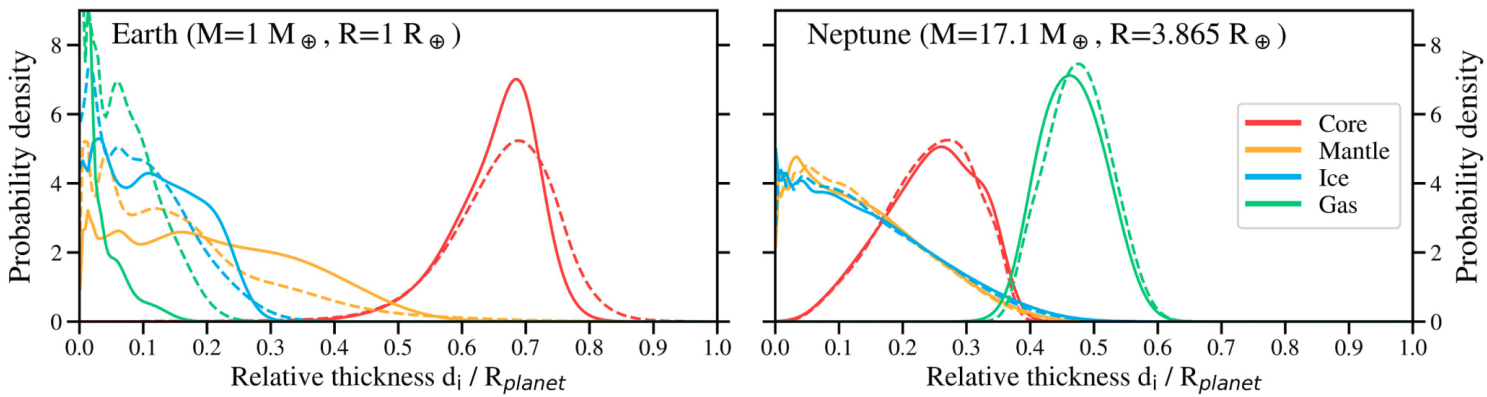
- A dataset of artificial planets is built from physics-based simulations.
- A MDN model mapping 'observed' variables (mass/radius/k_2) to 'hidden' variables (composition) is trained on the dataset [1].



Source: Baumeister et al. (2020), ML Inference of the Interior Structure of Low-mass Exoplanets

MDNs for Inferring Planets Composition

- The MDN model can be verified on planets for which the interior structure is known (e.g. planets in the solar system).



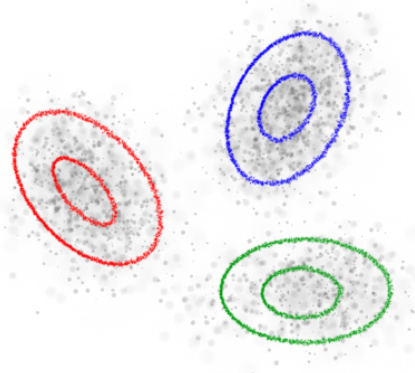
Source: Baumeister et al. (2020), ML Inference of the Interior Structure of Low-mass Exoplanets

- The MDN model can be used to infer the composition of less known exoplanets from a limited number of observations.

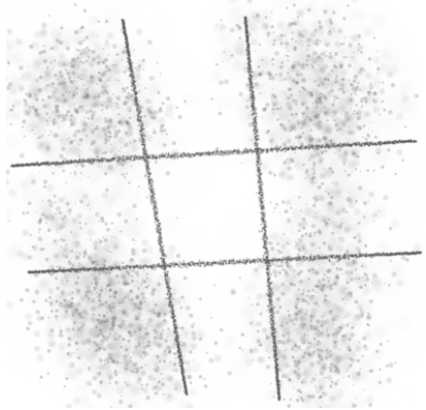
From Mixtures to Boltzmann Machines

- Gaussian mixture models (GMMs) capture local regions of high density but is not efficient for distributions that are shaped by global effects.
- Boltzmann machines (BMs) describe the probability function as a product of factors. Each factor captures a global effect in the distribution.

GMM



RBM



Conditional RBM

A conditional restricted Boltzmann machine [6] is a system of binary variables composed of an input $\boldsymbol{x}$, and output $\boldsymbol{y}$ and a latent state $\boldsymbol{h}$

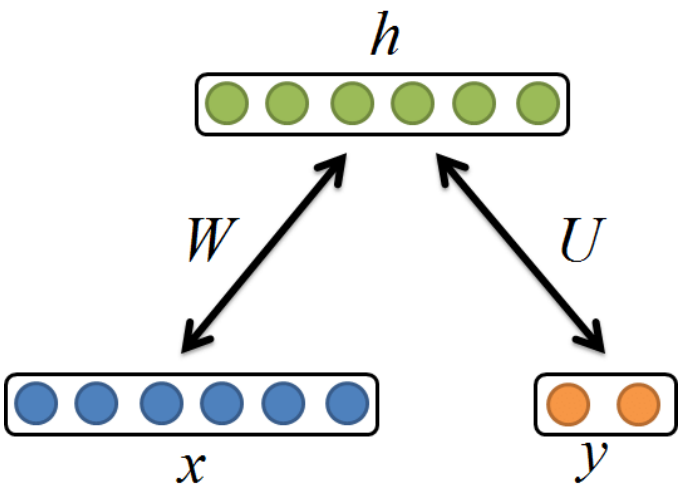


Image from Jing et al. (2014) Ensemble of ML algorithms for cognitive and physical speaker load detection

- It is governed by an energy function

$$E(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{y}) = -\boldsymbol{x}^\top W \boldsymbol{h} - \boldsymbol{y}^\top U \boldsymbol{h}$$

- The system associates to each joint configuration the probability

$$p(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{y}) = Z^{-1} \exp(-E(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{y}))$$

where Z is a normalization term.

Conditional RBM, Marginalization

Marginalization can be easily achieved in a CRBM:

$$p(\boldsymbol{x}, \boldsymbol{y}) = Z^{-1} \exp(-F(\boldsymbol{x}, \boldsymbol{y}))$$

where

$$F(\boldsymbol{x}, \boldsymbol{y}) = -\sum_{k=1}^{K} \log(1 + \exp(W_{:k}^{\top} \boldsymbol{x} + U_{:k}^{\top} \boldsymbol{y}))$$

is called the free energy and can be interpreted as a two-layer neural network. The lower the free energy the more likely the joint configuration $(\boldsymbol{x}, \boldsymbol{y})$. Hence, structured prediction can be performed as:

$$\boldsymbol{y}|\boldsymbol{x} = \arg\min_{\boldsymbol{y}} F(\boldsymbol{x}, \boldsymbol{y})$$

Training a Conditional RBM

A common training procedure is to maximize the data likelihood

$$\ell = \frac{1}{N} \sum_{n=1}^{N} \log p(\boldsymbol{x}^n, \boldsymbol{y}^n).$$

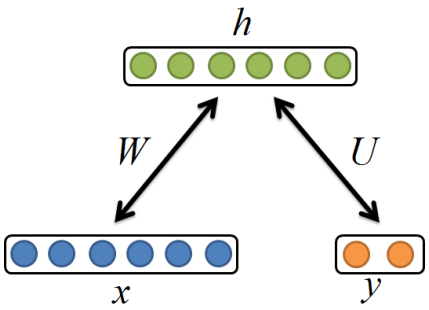
Its gradient is given by:

$$\frac{\partial \ell}{\partial W_{ik}} = \mathbb{E}_{\hat{p}(\boldsymbol{x},\boldsymbol{y})}\left[x_i \cdot \operatorname{sigm}(W_{:k}^\top \boldsymbol{x})\right] - \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\left[x_i \cdot \operatorname{sigm}(W_{:k}^\top \boldsymbol{x})\right]$$

$$\frac{\partial \ell}{\partial U_{jk}} = \mathbb{E}_{\hat{p}(\boldsymbol{x},\boldsymbol{y})}\left[y_i \cdot \operatorname{sigm}(U_{:k}^\top \boldsymbol{y})\right] - \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\left[y_i \cdot \operatorname{sigm}(U_{:k}^\top \boldsymbol{y})\right]$$

- The left hand side is a data expectation.
- The right hand side is an expectation over the probability distribution $p(\boldsymbol{x}, \boldsymbol{y})$ implemented by the CRBM. The latter can be approximated using alternate Gibbs sampling (i.e. iteratively sampling from $p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{h})$ and $p(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{y})$).

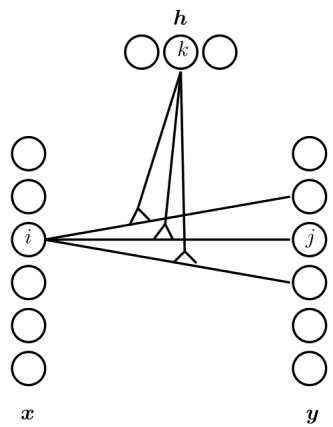
CRBM Variant: Product Interactions



- Standard CRBM

$$E(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{y}) = -\boldsymbol{x}^\top W \boldsymbol{h} - \boldsymbol{y}^\top U \boldsymbol{h}$$
$$= -\sum_{ik} x_i h_k W_{ik} - \sum_{jk} y_j h_k U_{jk}$$

Input and output contribute additively to the energy.



- Higher-order factored CRBM [5]

$$E(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{y}) = -\sum_{ijkf} x_i y_j h_k W_{if} U_{jf} V_{kf}$$

Input and output interact in a multiplicative manner.

Inspecting the CRBM (input filters W)

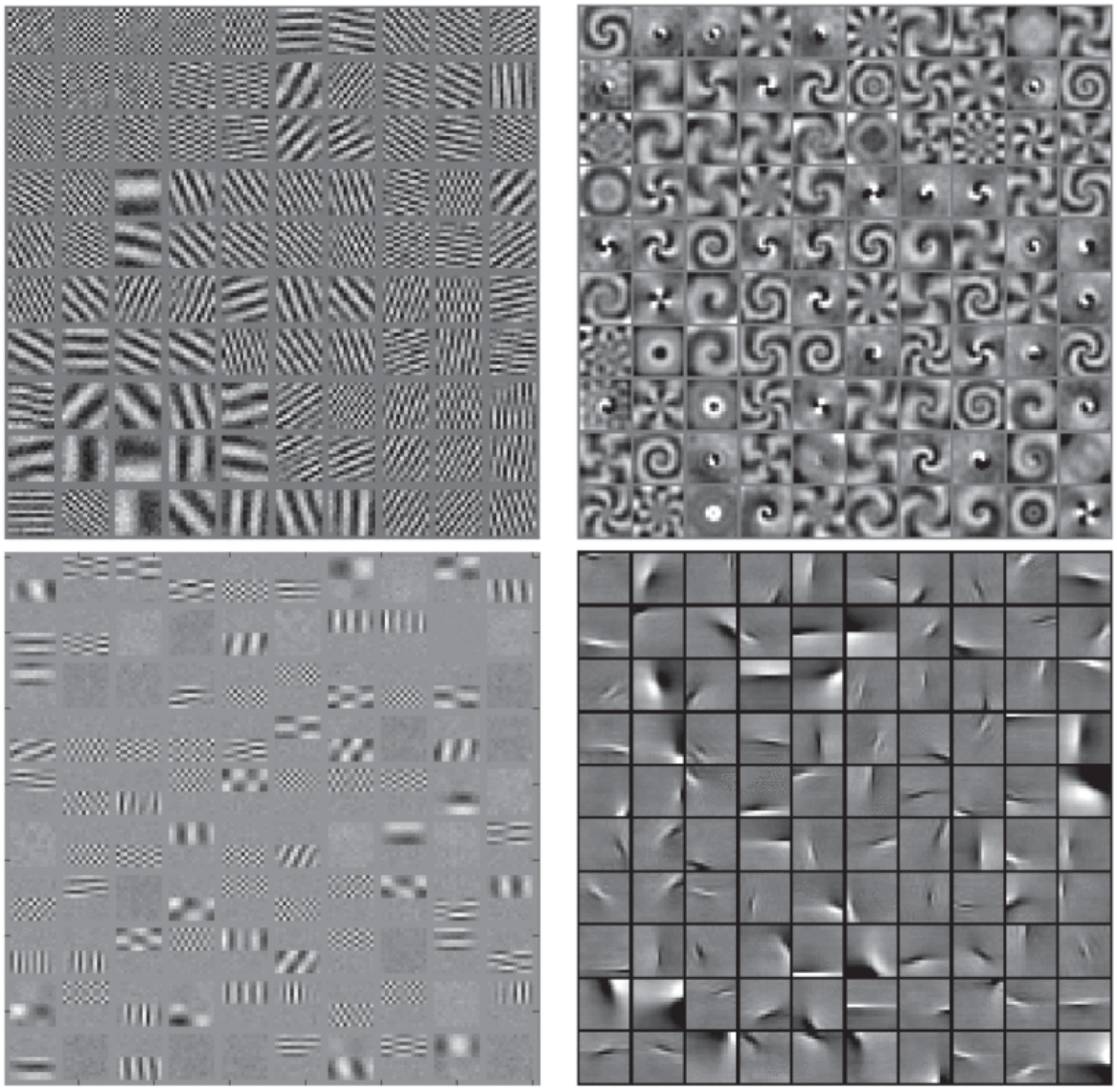


Image from Memisevic et al. (2013) Learning to Relate Images

Inspecting the CRBM (output filters U)

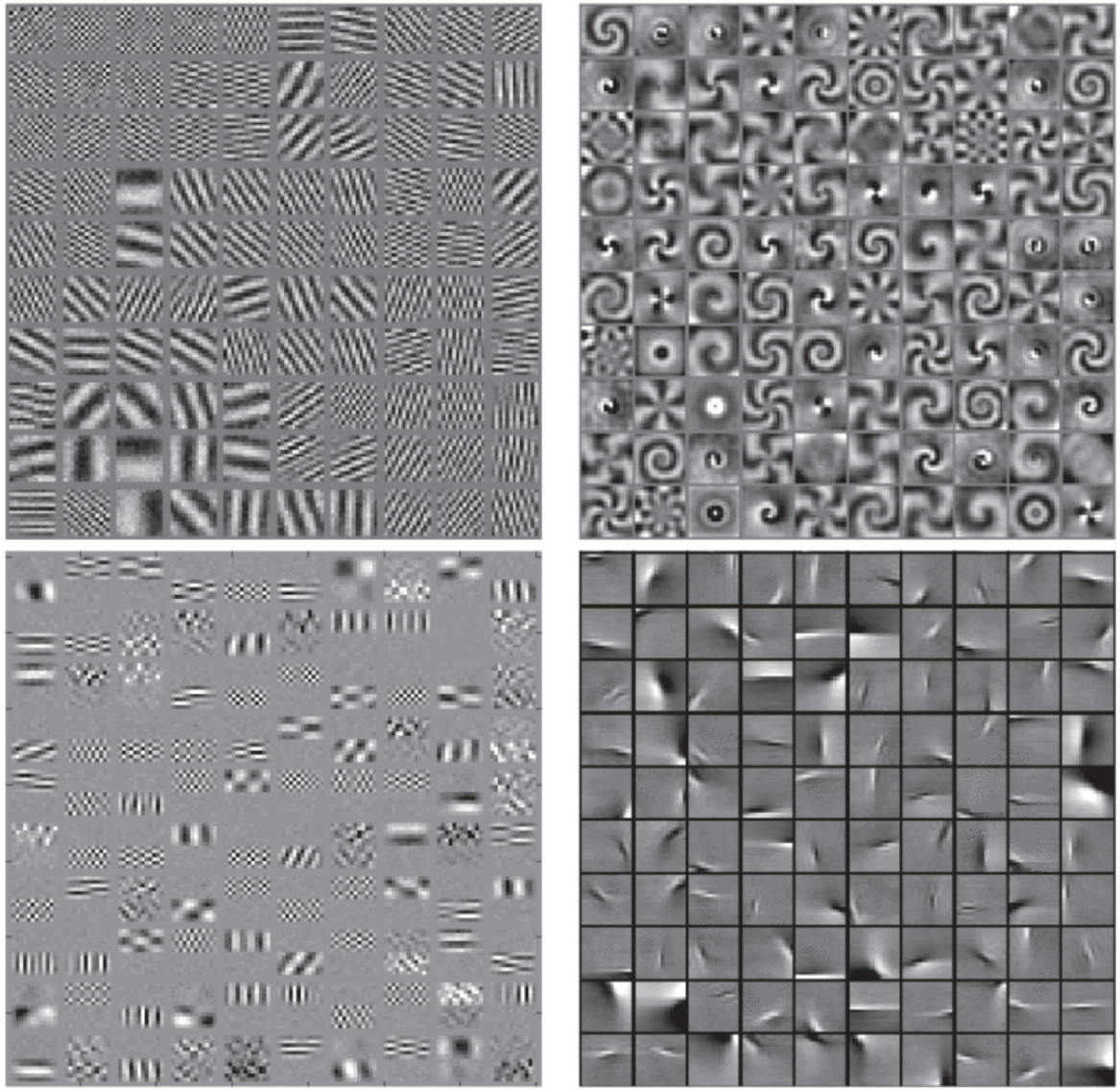
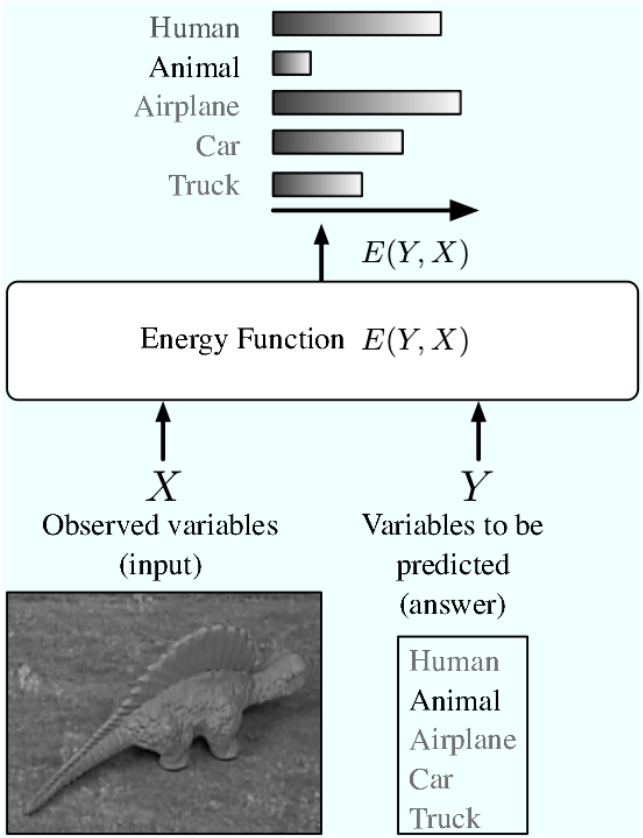


Image from Memisevic et al. (2013) Learning to Relate Images

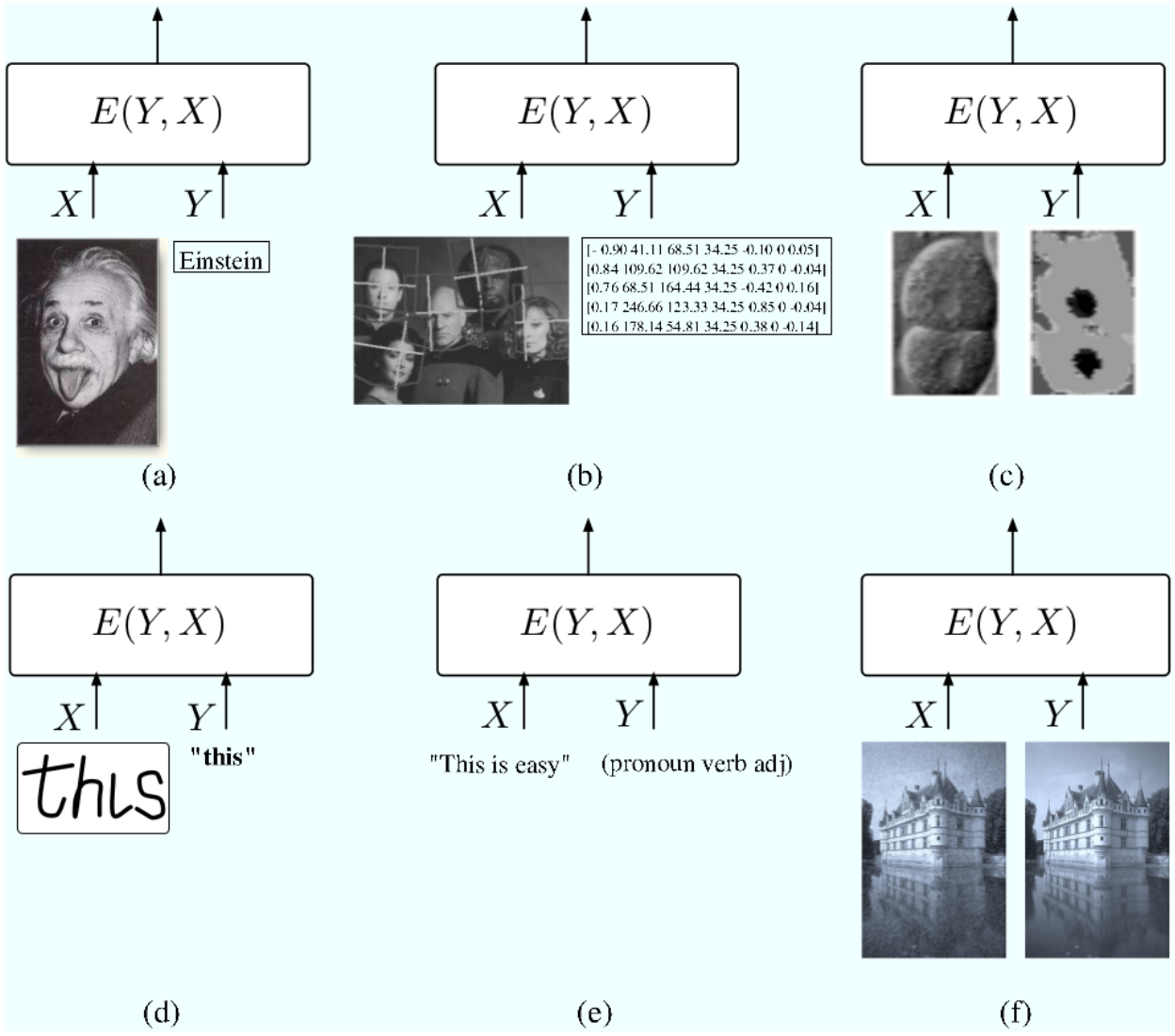
General Framework: Energy-Based Learning



- Energy-based learning [4] is a general framework for performing structured predictions with neural networks.
- Like for kernel-based structured prediction, it associates to each input-output pair a score.
- The predicted output is the one that yields the highest score (lowest energy).
- Most of the images in the following slides are from the paper:

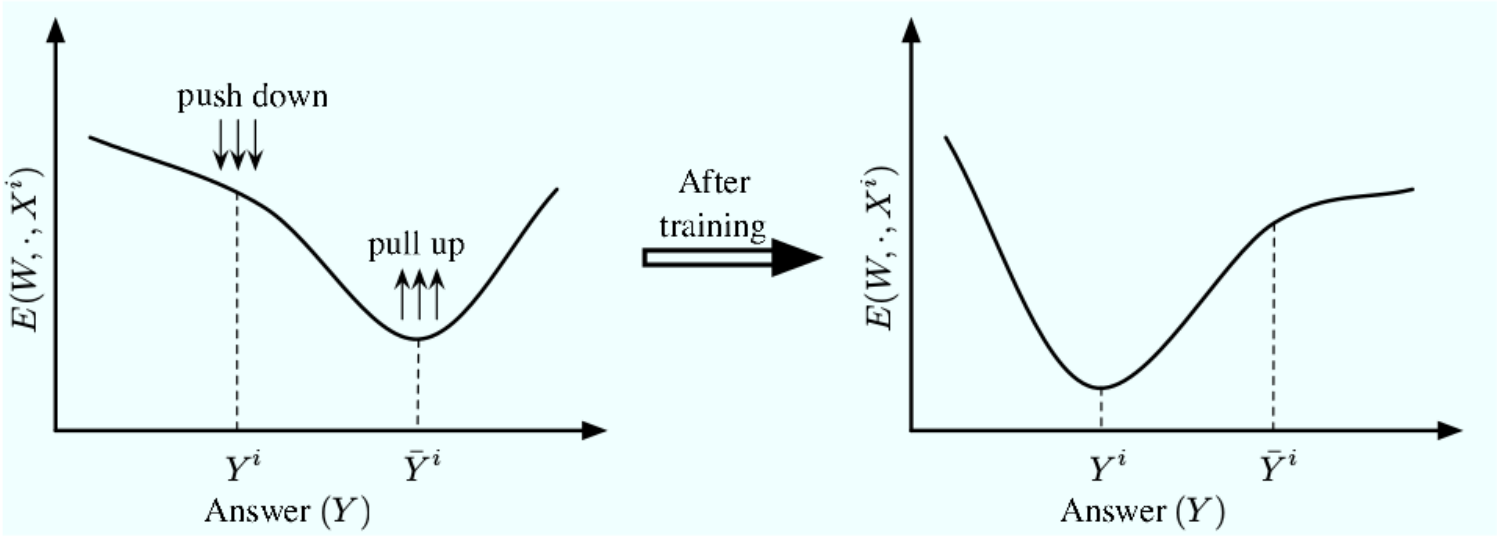
  LeCun et al. (2006) A Tutorial on Energy-Based Learning

Applications of Energy-Based Learning



Energy-Based Learning

The model can be trained using a push/pull approach



Example: Generalized perceptron loss

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$$

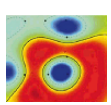
Energy-Based Learning

Examples of Loss Functions:

Loss (equation #)	Formula	Margin
energy loss (6)	$E(W, Y^i, X^i)$	none
perceptron (7)	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge (11)	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log (12)	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0

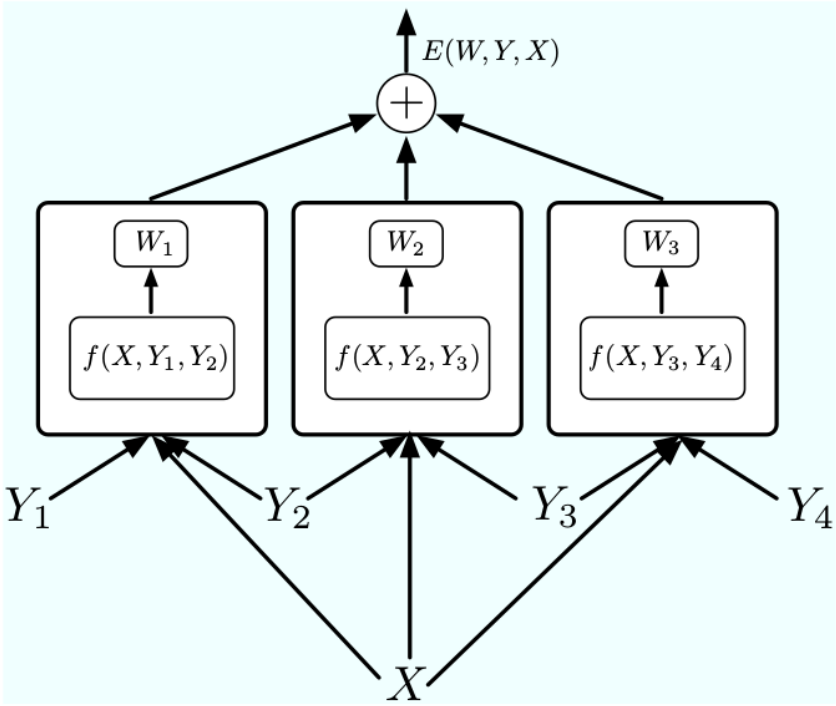
- Margin is an important component to ensure that the learned model generalizes well to test data.

Question: How to find the contrastive examples $\bar{Y}^i$ for which the loss function is high, i.e. examples with low energy?



Finding Contrastive Examples

For certain tasks where the output is sequential (e.g. segmentation) the model can be structured in a way that makes finding the best $\bar{Y}^i$ easy.



Key steps:

- Define a set of functions involving only adjacent items in the sequence.
- The best sequences $\bar{Y}^i$ can then be obtained using dynamic programming (cf. lecture on kernel-based structured prediction).

Finding Contrastive Examples

- For problems where dynamic programming is not applicable, we can still train a 'generator network' [3] to produce good contrastive examples.
- The generator network G_θ is trained in competition with the structured output objective, e.g. for the log-loss:

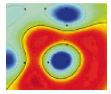
$$\min_W \max_\theta \; \log\left(1 + \exp\left(E_W(Y, X) - E_W(G_\theta(X), X)\right)\right)$$

Implementation trick: solve with a single optimizer

$$\min_{W,\theta} \; \log\left(1 + \exp\left(E_W(Y, X) - E_W(2G_\theta(X) - G_\theta(X), X)\right)\right)$$

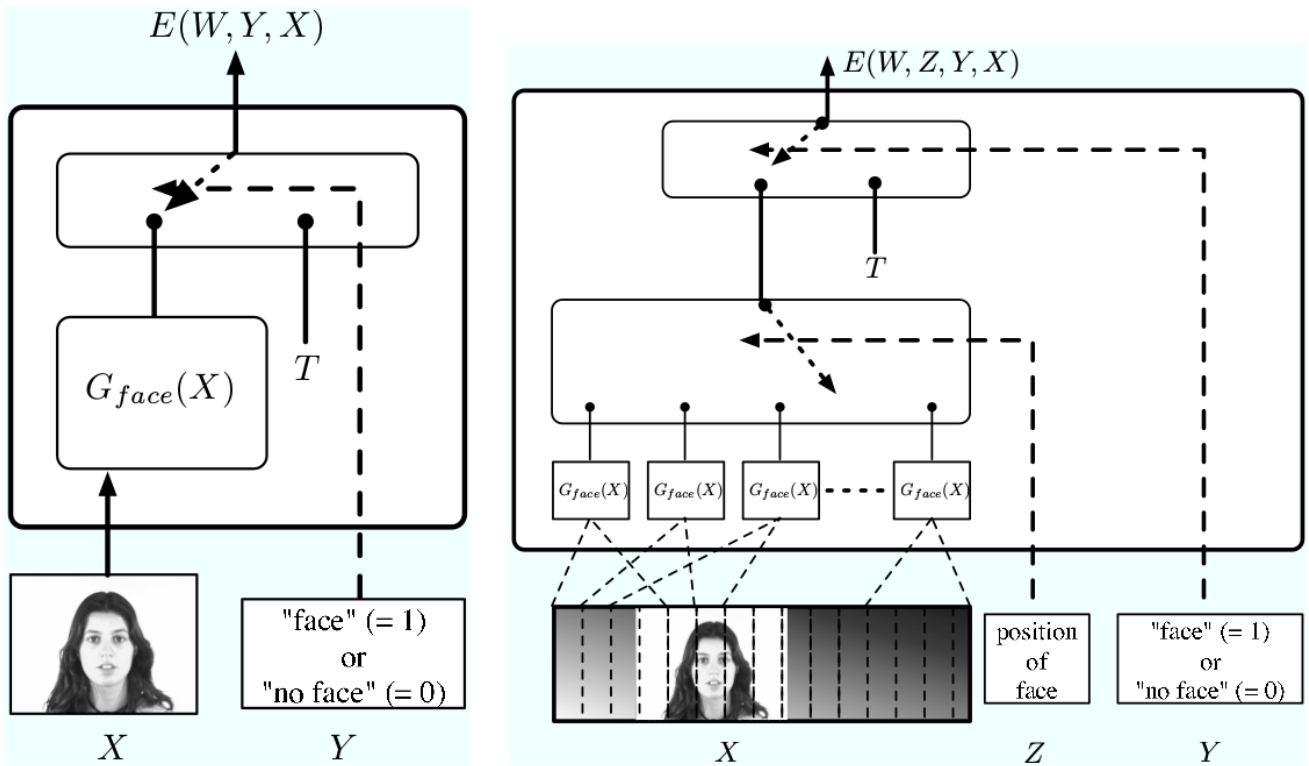
and detach the term in brown from the differentiation graph.

- We will use this method in the programming homework.



Adding Latent Variables

Useful to account for unknown factors (e.g. position of the face).



Latent variable can be marginalized as $E(W, X, Y) = \min_{Z \in \mathcal{Z}} E(W, Z, X, Y)$, potentially using dynamic programming.

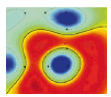
Structured Prediction: Kernels vs. Networks

Advantages of *Kernel-Based* Structured Prediction

- Margin is maximized in a well-defined feature space $\phi(\boldsymbol{x})$.
- Learning algorithm is convex.

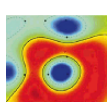
Advantage of *Neural Network-Based* Structured Prediction

- We have more flexibility in letting the model extract the representation it needs for the task.
- The output space $\mathcal{Y}$ can be continuous (e.g. images) and inference of $y|x$ can be done via gradient descent.



Summary

- Like kernel machines, neural networks need to be adapted to be able to perform structured predictions.
- Simple methods for structured output learning include mixture density networks and conditional RBMs.
- More general structured predictions (e.g. sequences, trees, etc.) can be achieved within the flexible framework of energy-based learning.



References I

P. Baumeister, S. Padovan, N. Tosi, G. Montavon, N. Nettelmann, J. MacKenzie, and M. Godolt.

Machine-learning inference of the interior structure of low-mass exoplanets.

The Astrophysical Journal, 889(1):42, Jan. 2020.

C. M. Bishop.

Neural Networks for Pattern Recognition.

Oxford University Press, Inc., USA, 1995.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio.

Generative adversarial nets.

In *NIPS*, pages 2672–2680, 2014.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang.

A tutorial on energy-based learning.

MIT Press, 2006.

R. Memisevic and G. E. Hinton.

Learning to represent spatial transformations with factored higher-order boltzmann machines.

Neural Computation, 22(6):1473–1492, 2010.

V. Mnih, H. Larochelle, and G. E. Hinton.

Conditional restricted boltzmann machines for structured output prediction.

In *UAI*, pages 514–522. AUAI Press, 2011.

I. Sutskever, O. Vinyals, and Q. V. Le.

Sequence to sequence learning with neural networks.

In *NIPS*, pages 3104–3112, 2014.