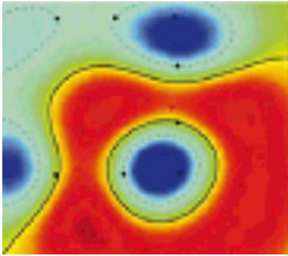
WiSe 2024/25

Machine Learning 1/1-X

Technische Universität Berlin



Lecture 14

Product of Experts

Outline

Covered content:

- Products of Experts (PoE)
- Restricted Boltzmann Machine (RBM)
- Structure of an RBM
- RBM learning algorithms
- Application of RBMs

Reference:

- Hinton, Geoffrey E. (2002). "Training Products of Experts by Minimizing Contrastive Divergence" (PDF). Neural Computation. 14 (8): 1771–1800

Recap: Gaussian Mixture Models (1)

Gaussian Mixture Model represents a (multimodal) probability density function as a weighted sum of K Gaussian components

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

where

- K is the number of the individual components
- $\boldsymbol{\theta} = \{\alpha_1, ..., \alpha_K, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, \Sigma_1, ..., \Sigma_K\}$ are the model parameters.
- α_k with $\alpha_k \geqslant 0$, $\sum_k \alpha_k = 1$ are the mixing coefficients representing the weight of each Gaussian component. During sampling, α_k gives the probability that $\boldsymbol{x}$ has been generating by the k-th component.
- $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)$ is the Gaussian distribution for component k, defined as

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

Recap: Gaussian Mixture Models (2)

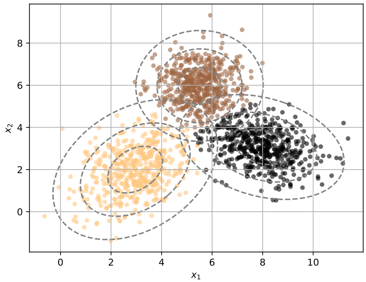
- **Gaussian Mixture Model** are an example of latent variable models. For a latent variable $y \in \{1, ..., K\}$

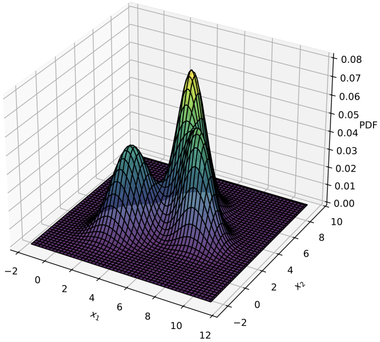
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k) = \sum_{k=1}^{K} \underbrace{p(y = k)}_{\alpha_k} \underbrace{p(x|y = k)}_{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)}$$

  where

- Given observed data points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$, we can use the **Expectation Maximization** algorithm for **Maximum Likelihood Estimation** of the corresponding parameters $\boldsymbol{\theta} = \{\alpha_1, ..., \alpha_K, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, \Sigma_1, ..., \Sigma_K\}$.
- Gaussian Mixture Models (GMMs) can approximate any smooth probability distribution to an arbitrary degree of accuracy.

Recap: Gaussian Mixture Models (3)





Product of Experts

Given K experts, each represented by an (unnormalized) probability density function $g_k(\boldsymbol{x}|\boldsymbol{\theta}_k)$, the combined model

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \prod_{k=1}^{K} g_k(\boldsymbol{x}|\boldsymbol{\theta}_k)$$

is called **Product of Experts**, where

- $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K\}$ are the model parameters
- K is the number of the individual experts
- normalizing constant $\mathcal{Z}$ is the partition function

$$\mathcal{Z} = \int \prod_{k=1}^{K} g_k(\boldsymbol{x}|\boldsymbol{\theta}_k) d\boldsymbol{x}$$

Example 1: Product of Gaussians (1)

For $\boldsymbol{x} \in \mathbb{R}^d$, consider the experts $p_i(x_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ to be univariate Gaussians specialized on a particular input dimension $i \in \{1, ..., d\}$ according to

$$p_i(\boldsymbol{x}|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}\right)$$

The product of experts (see next slide for derivation) is given as:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) &= \prod_{i=1}^{d} p_i(\boldsymbol{x}|\mu_i, \sigma_i^2) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right) \end{aligned}$$

在 $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ 的情况下，即一个特征之间没有相关性的多元高斯分布。通常，任何有限数量的（多元）独立高斯分布的乘积也是一个高斯分布，尽管变量之间可能存在相关性。由于结果分布是高斯分布，专家乘积（与混合模型不同）无法近似任意的平滑分布。然而，由于模型参数数量的减少，我们获得了效率上的提升。

with $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$, i.e. a multivariate Gaussian distribution without correlation between features. Generally, any finite product of (multivariate) independent Gaussians is also a Gaussian with potentially correlated variables. Since the resulting distribution is a Gaussian, the product of experts (unlike mixture models) cannot approximate arbitrary smooth distribution. However, we gain efficiency due to the reduction in the number of model parameters.

Example 1: Product of Gaussians (2)

For $\boldsymbol{\mu} := (\mu_1, ..., \mu_d)$, $\Sigma := \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ and $\Sigma^{-1} = \mathrm{diag}(\frac{1}{\sigma_1^2}, \ldots, \frac{1}{\sigma_d^2})$ we consider the following derivation of the exponential term:

$$\begin{aligned}\prod_{i=1}^{d} \exp\Big(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\Big) &= \exp\Big(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i-\mu_i)^2}{\sigma_i^2}\Big) \\ &= \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right),\end{aligned}$$

where $\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = (\frac{x_1-\mu_1}{\sigma_1^2}, ..., \frac{x_d-\mu_d}{\sigma_d^2})^\top$.
Similarly, for the remaining terms we get:

$$\prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_i^2}} = \frac{1}{\sqrt{(2\pi)^d \prod_{i=1}^{d} \sigma_i^2}} = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}}$$

In particular, $\mathcal{Z} = 1$.

Example 2: Product of Gaussian Mixtures

Let each expert be a mixture of univariate Gaussian distributions

$$p_i(\boldsymbol{x}|\boldsymbol{\theta}_i) = \sum_{k=1}^{K} \alpha_{ik} \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} \exp\left(-\frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2} \right)$$

The product of experts (PoE) can be developed as:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{\theta}) &= \frac{1}{\mathcal{Z}} \prod_{i=1}^{d} \sum_{k=1}^{K} \alpha_{ik} \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} \exp\left(-\frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2} \right) \\ &= \sum_{k_1=1}^{K} \cdots \sum_{k_d=1}^{K} \underbrace{\left(\frac{1}{\mathcal{Z}} \prod_{i=1}^{d} \alpha_{ik_i} \right)}_{\text{mix. coef.}} \underbrace{\left(\prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{ik_i}^2}} \exp\left(-\frac{(x_i - \mu_{ik_i})^2}{2\sigma_{ik_i}^2} \right) \right)}_{\text{multivar. Gaussian } \mathcal{N}(\boldsymbol{\mu}_{k_1,\dots,k_d}, \Sigma_{k_1,\dots,k_d})} \end{aligned}$$

yielding a mixture of *exponentially* many (K^d) multivariate Gaussians with uncorrelated variables, where $\boldsymbol{\mu}_{k_1,\dots,k_d} = (\mu_{1k_1}, \dots, \mu_{dk_d})^\top$ and $\Sigma_{k_1,\dots,k_d} = \mathrm{diag}(\sigma_{1k_1}^2, \dots, \sigma_{dk_d}^2)$ Therefore, a PoE can encode (some!) mixture models by using fewer parameters. In general, PoE is less expressive.

Example 3: Product of t-Student Distributions

Define H experts to be (a special case of) t-Student distributions in some projected space $z = \boldsymbol{w}_i^\top \boldsymbol{x}$ with $\|\boldsymbol{w}_i\| = 1$:

$$p_i(\boldsymbol{x}|\boldsymbol{\theta}_i) = \frac{1}{\alpha_i + (\boldsymbol{w}_i^\top \boldsymbol{x})^2}$$

The resulting product of experts

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \prod_{j=1}^{H} \frac{1}{\alpha_i + (\boldsymbol{w}_i^\top \boldsymbol{x})^2}$$

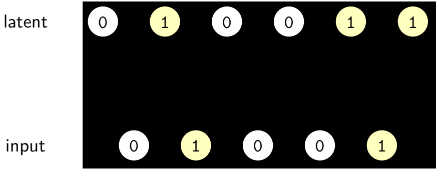
produces a non-Gaussian multivariate distribution, which can be useful to model e.g. image or speech data. This PoE has connections to other analyses, e.g. independent component analysis (ICA).

Mixture Models vs. Product of Experts

Aspect	Product of Experts (PoE)	Mixture Models
Combination Rule	Multiply distributions ($\prod$)	Add distributions ($\sum$)
Focus	Sharpens the distribution by focusing on intersections of experts.	Broadens the distribution by summing over components.
Interpretation	Experts act collaboratively to explain the data.	Components act competitively to explain the data.
Modeling	Captures complex interactions and dependencies.	Captures multi-modal distributions or clusters.
Normalization	Requires computing a potentially complex normalizer $\mathcal{Z}$.	Weights π_i handle normalization directly.

方面	专家乘积（PoE）	混合模型
组合规则	分布相乘（∏）	分布相加（∑）
关注点	通过关注专家的交集来锐化分布。	通过对组件的求和来拓宽分布。
解释	专家协作解释数据。	组件竞争解释数据。
建模	捕捉复杂的交互和依赖关系。	捕捉多模态分布或聚类。
归一化	需要计算可能复杂的归一化因子 $\mathcal{Z}$。	权重 π_i 直接处理归一化。

The Restricted Boltzmann Machine



The restricted Boltzmann machine (RBM) is a joint probability model defined over input features $\boldsymbol{x} \in \{0,1\}^d$ **and** latent variables $\boldsymbol{h} \in \{0,1\}^H$.

$$p(\boldsymbol{x}, \boldsymbol{h} | \boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \exp\Big(\sum_{i=1}^{d} \sum_{j=1}^{H} x_i w_{ij} h_j + \sum_{j=1}^{H} b_j h_j \Big)$$

The parameter w_{ij} can be interpreted as the connection strength between input feature x_i and latent variable h_j. The larger w_{ij} the stronger x_i and h_j co-activate.

The Restricted Boltzmann Machine (PoE View)

当 RBM（受限玻尔兹曼机）在其隐藏单元上被边缘化时，它具有专家乘积（Product of Experts, PoE）的结构

Connection between RBM and PoE

The RBM, when marginalized over its hidden units, has the structure of a Product of Experts with $g(\boldsymbol{x}|\boldsymbol{\theta}_j) = (1 + \exp(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j))$.

Proof:

$$p(\boldsymbol{x}|\theta) = \sum_{\boldsymbol{h}\in\{0,1\}^H} p(\boldsymbol{x}, \boldsymbol{h}|\theta) = \sum_{\boldsymbol{h}\in\{0,1\}^H} \frac{1}{\mathcal{Z}} \exp\left(\sum_{i=1}^{d}\sum_{j=1}^{H} x_i w_{ij} h_j + \sum_{j=1}^{H} b_j h_j\right)$$

$$= \sum_{\boldsymbol{h}\in\{0,1\}^H} \frac{1}{\mathcal{Z}} \exp\left(\sum_{j=1}^{H}((\sum_{i=1}^{d} x_i w_{ij}) + b_j)h_j\right)$$

$$= \sum_{\boldsymbol{h}\in\{0,1\}^H} \frac{1}{\mathcal{Z}} \exp\left(\sum_{j=1}^{H}(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j)\cdot h_j\right) = \frac{1}{\mathcal{Z}} \sum_{\boldsymbol{h}\in\{0,1\}^H} \prod_{j=1}^{H} \exp((\boldsymbol{w}_j^\top \boldsymbol{x} + b_j)\cdot h_j)$$

$$= \frac{1}{\mathcal{Z}} \prod_{j=1}^{H} \sum_{h_j\in\{0,1\}} \exp((\boldsymbol{w}_j^\top \boldsymbol{x} + b_j)\cdot h_j)$$

$$= \frac{1}{\mathcal{Z}} \prod_{j=1}^{H} (1 + \exp(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j))$$

Interpreting the RBM Experts

The experts

$$g(\boldsymbol{x}|\boldsymbol{\theta}_j) = 1 + \exp(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j)$$

forming the RBM implement two behaviors:

- $\boldsymbol{w}_j^\top \boldsymbol{x} + b_j > 0 : g(\boldsymbol{x}|\boldsymbol{\theta}_j) \gg 1$: the example $\boldsymbol{x}$ is in the area of competence of the expert and the latter speaks in favor of $\boldsymbol{x}$.
- $\boldsymbol{w}_j^\top \boldsymbol{x} + b_j < 0 : g(\boldsymbol{x}|\boldsymbol{\theta}_j) \approx 1$: the example $\boldsymbol{x}$ is outside the expert's area of competence, and the expert 'withdraws', i.e. multiplies by 1.

This double behavior is important to implement the nonlinearity.

- 当 $\mathbf{w}_j^T\mathbf{x} + b_j > 0$ 时:

$$g(\mathbf{x}|\theta_j) \gg 1$$

  这意味着样本 $\mathbf{x}$ 位于该专家的能力范围内，该专家对 $\mathbf{x}$ 表达支持。

- 当 $\mathbf{w}_j^T\mathbf{x} + b_j < 0$ 时:

$$g(\mathbf{x}|\theta_j) \approx 1$$

  这意味着样本 $\mathbf{x}$ 位于该专家的能力范围之外，该专家“撤回”自己的影响，即仅作为单位因子存在（乘以 1）。

这种双重行为对于实现非线性至关重要。

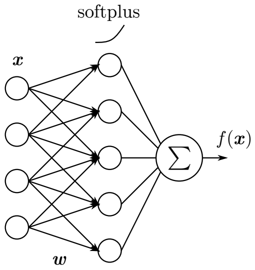
RBM as a Neural Network (1)

The product of expert (PoE) model

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \prod_{j=1}^{H} (1 + \exp(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j))$$

can be rewritten as:

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}) = \underbrace{\sum_{j=1}^{H} \underbrace{\log(1 + \exp(}_{\text{softplus}} \boldsymbol{w}_j^\top \boldsymbol{x} + b_j))}_{f_{\boldsymbol{\theta}}(\boldsymbol{x})} - \log \mathcal{Z}$$



where $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be interpreted as a neural network. Here, the **softplus** operation implements the smooth version of the ReLU: $x \mapsto \max(0, x)$.

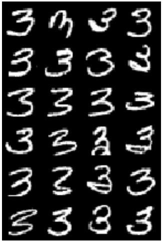
相关

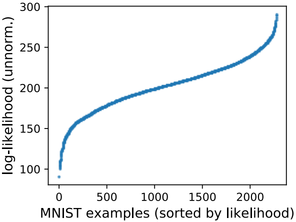
Note: The neural network can predict which examples $\boldsymbol{x}$ are more probable relative to other examples, but not the actual probability score, because $\log \mathcal{Z}$ is difficult to compute.

RBM as a Neural Network (2)

RBM trained on MNIST digits (only digits “3” with 100-125 foreground pixels).

2% least likely





2% most likely

Observations:

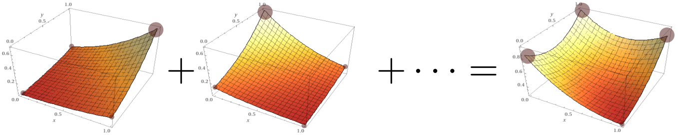
- Digits with anomalies are predicted by the RBM to be less likely than clean digits.

RBM as a Neural Network (3)

Universal approximation

The RBM is an universal approximator of data distributions in the binary input space $\{0,1\}^d$.

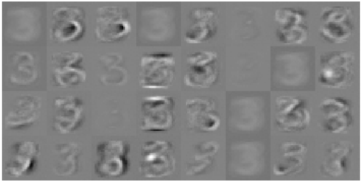
"Proof" by construction: Add a softplus neuron pointing to each corner of the hypercube. Scale the weight $\boldsymbol{w}_j$ according to the probability of each corner, and choose the bias accordingly.



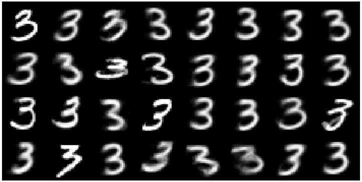
Note: This construction requires exponentially many neurons. In practice, each expert θ_j captures more than a single corner of the hypercube (i.e. learn general features).

RBM as a Neural Network (4)

RBM weights $(\boldsymbol{w}_j)_{j=1}^{H}$



K-Means centroids



Observation

- In the RBM, experts $(\boldsymbol{w}_j)_j$ encode only part of the digit (e.g. a stroke). The expert therefore contributes to make all data containing that particular stroke more likely.
- The experts that the RBM extracts can be used for transfer learning, e.g. reused to predict different digits/characters.
- K-means centroids are much more localized, and consequently less transferable.

- 在 RBM 中，专家（w_j）只编码数字的一部分（例如某个笔画）。因此，该专家的作用是使所有包含该特定笔画的数据更可能出现。
- RBM 提取的专家可以用于**迁移学习（transfer learning）**，例如可以用于预测不同的数字或字符。
- K-Means 质心更加局部化，因此**迁移性较差。**

Learning an RBM

Recap: mixture model and EM bound (ELBO)

The mixture model can be optimized by finding the optimum of a lower-bound at each iteration

$$\log p(\mathcal{D}|\theta) \geq \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_n)} \left[\log \frac{p(\boldsymbol{x}_n, \boldsymbol{z}|\theta)}{q(\boldsymbol{z}|\boldsymbol{x}_n)} \right] = \sum_{n=1}^{N} \sum_{k=1}^{K} \log \left(\frac{\alpha_k p(\boldsymbol{x}_n|\theta_k)}{\beta_k} \right) \beta_k$$

where N is the number of data points, and $(\alpha_k)_k$ and $(\beta_k)_k$ are distributions over the K mixture elements.

Question: Can the same EM approach be used for Product of Experts?

$$\log p(\mathcal{D}|\theta) = \sum_{n=1}^{N} \left[f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - \log \mathcal{Z} \right]$$

Answer: No, the expression cannot be bounded in the same way as for the mixture model (it has a different structure).

Gradient of the RBM

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \prod_{j=1}^{H} (1 + \exp(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j))$$

can be rewritten as:

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}) = \underbrace{\sum_{j=1}^{H} \underbrace{\log(1 + \exp(}_{\text{softplus}} \boldsymbol{w}_j^\top \boldsymbol{x} + b_j))}_{f_{\boldsymbol{\theta}}(\boldsymbol{x})} - \log \mathcal{Z}$$

Idea: Although EM is not applicable, we can still compute the gradient of the log-likelihood and perform gradient descent.

$$\begin{aligned}
\nabla_\theta \log p(\mathcal{D}|\theta) &= \nabla_{\boldsymbol{\theta}} \sum_{n=1}^{N} \left[f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - \log \mathcal{Z} \right] \\
&= \nabla_{\boldsymbol{\theta}} \sum_{n=1}^{N} \left[f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - \log \sum_{\boldsymbol{x} \in \{0,1\}^d} \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x})) \right] \\
&= \sum_{n=1}^{N} \left[\nabla_\theta f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - \frac{\sum_{\boldsymbol{x} \in \{0,1\}^d} \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x})) \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})}{\sum_{\boldsymbol{x} \in \{0,1\}^d} \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x}))} \right] \\
&= \sum_{n=1}^{N} \nabla_\theta f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - N \cdot \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \right]
\end{aligned}$$

The gradient is a difference between a data-dependent and a model-dependent (i. e. data-independent) term.

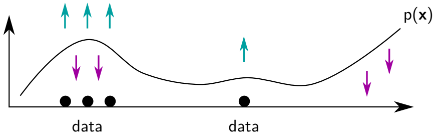
RBM Update Rule

Based on the gradient calculation above, we can build the update rule:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \gamma \cdot \left(\frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] \right)$$

Interpretation:

- The first term favours the model which makes the observed data more likely.
- The second term makes everything that the model considers likely, less likely.
- The training procedure terminates when we reach some equilibrium.



- **第一项（first term）** 使模型倾向于增加观察数据的概率。
- **第二项（second term）** 使模型中已经认为可能性较高的样本概率降低。
- **训练过程在达到某种平衡（equilibrium）时终止。**

Computation of the RBM update rule

左侧项（left term） 计算简单：可以通过 $f_\theta(x)$ 进行反向传播（backprop），并对所有数据点取平均。

右侧项（right term） 计算复杂：因为 $p(x|\theta)$ 被定义在指数级状态空间上，因此难以直接求解。
解决方案是使用**从模型分布** $p(x|\theta)$ **采样的方法**来得到梯度的无偏估计。

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \gamma \cdot \left(\frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] \right)$$

Observations

- The left term is easy to compute (backprop in $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, averaged over all data points).
- The right term is more tricky, because $p(\boldsymbol{x}|\boldsymbol{\theta})$ is defined over exponentially many states. An unbiased approximation of the expected gradient can be obtained by generating a sample $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ from the model distribution $p(\boldsymbol{x}|\theta)$.

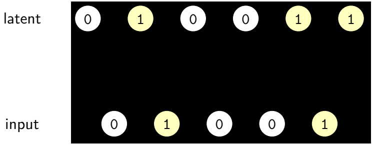
Question: How do we sample from $p(\boldsymbol{x}|\boldsymbol{\theta})$?

- *Idea:* Switch back to the 'classical' (i.e. non-PoE) view of the RBM, where we can use latent variables to ease sampling.

问题：如何从 $p(x|\theta)$ **进行采样？**

- **思路（Idea）**：
  - 回归到**经典的（classical，即非 PoE）** RBM 视角。
  - 在该视角下，我们可以利用**潜在变量（latent variables）** 来简化采样过程。

Recap: 'Classical' View of the RBM



The RBM is a probability model defined over input features $\boldsymbol{x} \in \{0,1\}^d$ and **and** latent variables $\boldsymbol{h} \in \{0,1\}^H$.

$$p(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \exp\left(\sum_{i=1}^{d} \sum_{j=1}^{H} x_i w_{ij} h_j + \sum_{j=1}^{H} b_j h_j \right)$$

The parameter w_{ij} can be interpreted as the connection strength between input feature x_i and latent variable h_j. The larger w_{ij} the stronger x_i and h_j co-activate.

Conditional Distributions in an RBM

抽样 $p(x)$ 和 $p(h)$ 是困难的，但当我们条件化 x 或 h 时，隐变量 h 可以被独立且容易地采样，同样，x 在条件化 h 时也可以被采样。

Sampling $p(\boldsymbol{x})$ and $p(\boldsymbol{h})$ is hard, however, hidden variables $\boldsymbol{h}$ can be sampled *easily* and *independently* when we condition on the state $\boldsymbol{x}$, and similarly for $\boldsymbol{x}$ conditioned on $\boldsymbol{h}$. Let $z_j = \sum_i x_i w_{ij} + b_j$. We proceed as:

$$p(\boldsymbol{h}|\boldsymbol{x},\theta) = \frac{\frac{1}{\mathcal{Z}}\exp\left(\sum_j z_j h_j\right)}{\sum_{\boldsymbol{h}} \frac{1}{\mathcal{Z}}\exp\left(\sum_j z_j h_j\right)} = \frac{\prod_j \exp(z_j h_j)}{\prod_j \sum_{h_j} \exp(z_j h_j)} = \prod_j \underbrace{\frac{\exp(z_j h_j)}{1+\exp(z_j)}}_{p(h_j|\boldsymbol{x},\theta)}$$

and we observe that, conditioned on $\boldsymbol{x}$, the latent variables $(h_j)_j$ can be sampled *easily* (Bernoulli distributions) and *independently* (hidden variables in RBM are conditionally independent given $\boldsymbol{x}$).

By symmetry of the RBM model, we get a similar result for the visible units conditioned on the latent variables, i.e. $p(x_i|\boldsymbol{h},\theta) = \exp(z_i h_i)/(1+\exp(z_i))$ with $z_i = \sum_j w_{ij} h_j$.

Block Gibbs Sampling

Observation: We know $p(h_j|\boldsymbol{x},\theta)$ and $p(x_i|\boldsymbol{h},\theta)$. But how do we sample $p(\boldsymbol{x}|\theta)$, which we need to compute the RBM gradient?

Block Gibbs Sampling: Start with some random input $\boldsymbol{x}^{(0)}$, then sample alternately:

$$\begin{aligned}
\boldsymbol{h}^{(0)} &\sim \left(p(h_j \mid \boldsymbol{x}^{(0)}, \theta)\right)_{j=1}^{H} \\
\boldsymbol{x}^{(1)} &\sim \left(p(x_i \mid \boldsymbol{h}^{(0)}, \theta)\right)_{i=1}^{d} \\
\boldsymbol{h}^{(1)} &\sim \left(p(h_j \mid \boldsymbol{x}^{(1)}, \theta)\right)_{j=1}^{H} \\
\boldsymbol{x}^{(2)} &\sim \left(p(x_i \mid \boldsymbol{h}^{(1)}, \theta)\right)_{i=1}^{d} \\
&\vdots
\end{aligned}$$

The procedure guarantees that $\boldsymbol{x}^{(\cdot)}$ converges in distribution to $p(\boldsymbol{x}|\boldsymbol{\theta})$, that is,

$$\lim_{n\to\infty} F_{X^{(n)}}(\boldsymbol{x}) = F_X(\boldsymbol{x})$$

where $F_X(\boldsymbol{x}) = P(X_1 \leqslant x_1, ..., X_d \leqslant x_d)$ is the cumulative distribution function.

Fast Approximations

In practice, Gibbs sampling can take a long time to converge. Instead, we can use fast approximations of converged Gibbs sampling.

Common approximation strategies:

- *Contrastive divergence (CD-k)*: Start from an existing data point, and perform k steps of alternate Gibbs sampling.
- *Persistent contrastive divergence (PCD)*: Start from the Gibbs sample $\boldsymbol{x}$ in the previous iteration of gradient descent, and perform one step of Gibbs sampling.

Application: RBM for Data Denoising

Suppose you receive an example $\boldsymbol{x}_{\text{noisy}}$, and would like to denoise it. A reconstruction of that digit can be obtained by mapping to the latent variables and projecting back:

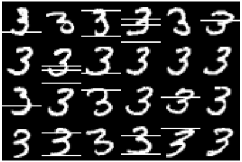
1. Projection on latent variables

$$\boldsymbol{h} = \text{sigm}(W\boldsymbol{x}_{\text{noisy}} + \boldsymbol{b})$$

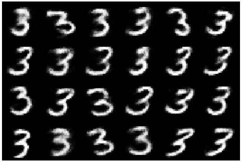
2. Backprojection on the input domain

$$\boldsymbol{x}_{\text{rec}} = \text{sigm}(W^\top \boldsymbol{h})$$

Before denoising



After denoising



Application: RBM for Representation Learning

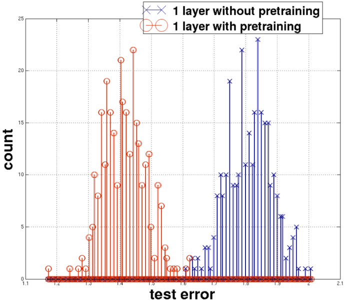
The RBM model can be used as a neural network initialization to achieve lower generalization error on some classification task.

- **Step 1:** Create a layer of features based on the RBM learned parameters:

$$\phi(\boldsymbol{x}) = \begin{pmatrix} \mathrm{sigm}(\boldsymbol{w}_1^\top \boldsymbol{x}) \\ \vdots \\ \mathrm{sigm}(\boldsymbol{w}_H^\top \boldsymbol{x}) \end{pmatrix}$$

- **Step 2:** Add a top layer that maps $\phi(\boldsymbol{x})$ to the class probabilities, and train the resulting neural network end-to-end using gradient descent.

MNIST example:



Source: Erhan et al. (2010) Why Does Unsupervised Pre-training Help Deep Learning?

Summary

- The Product of Experts is an unsupervised learning approach that is substantially different from Mixture Models.
- Product of experts such as the RBM are optimized by gradient descent (instead of EM).
- The RBM has several desirable properties: Simple gradient, block Gibbs sampler (quite fast).
- The RBM can be used for various tasks, e.g. probability modeling, data denoising, representation learning.