Advanced techniques in applied economics

Lecture 5: Latent variable models — from trees to neural nets

Piotr Zwiernik

upf. Universitat Pompeu Fabra Barcelona

Statistics, Probability and Machine Learning Research Group

B S E Barcelona School of Economics

Spring 2026

Latent structure as hidden economic mechanism

What changes today?

In the previous lectures, graphs described dependence among *observed* variables. Today we allow some important variables to be **hidden/latent**.

Why hidden variables?

- market-wide shocks are not directly observed,
- state of the economy is only visible through some indicators,
- consumers or firms belong to hidden segments.

Dimension reduction: PCA and related methods assume latent structure in the data.

Main question: Can a small number of latent variables explain complicated patterns in the data?

Main message: latent variables can turn a complicated observed dependence pattern into a simpler hidden structure.

Roadmap for today

1. Factor analysis and hidden common factors
2. Latent tree models and hierarchical structure
3. Restricted Boltzmann machines as a bridge to neural nets

When dependence is too complicated at the surface, look for a simpler hidden layer underneath.

Part 1: Factor analysis

Why economists care about factor models

Finance
A large collection of returns often moves because of a few common factors:

- market return,
- sectoral shocks,
- liquidity or volatility factors.

Applied micro and macro
Latent variables can represent

- ability,
- sentiment,
- business-cycle conditions,
- hidden household or firm types.

Factor model: many variables may comove because they are driven by the same hidden source.

For standard treatment, see Chapter 9 in Hastie, Tibshirani, and Friedman, *The Elements of Statistical Learning*.

The factor analysis model

The basic Gaussian factor model is

$$X = \mu + WZ + \varepsilon, \qquad Z \sim N_r(0, I_r), \qquad \varepsilon \sim N_m(0, \Psi), \qquad Z \perp\!\!\!\perp \varepsilon,$$

where Ψ is diagonal.

Interpretation

- $Z \in \mathbb{R}^r$ are the latent factors,
- $W \in \mathbb{R}^{m \times r}$ is the loading matrix,
- Ψ captures variable-specific noise not explained by the common factors.

Covariance form

$$\Sigma = WW^\top + \Psi.$$

So dependence is split into a **low-rank** common part and a **diagonal** idiosyncratic part.

A one-factor economic example

Suppose

$$X_i = \lambda_i F + \varepsilon_i, \qquad i = 1, \ldots, m,,$$

where F is a latent factor, and the ε_i are independent of each other and of F.

Observable restriction: For $i \neq j$,

$$\operatorname{cov}(X_i, X_j) = \lambda_i \lambda_j \operatorname{var}(F).$$

So the off-diagonal covariances have a multiplicative form. Equivalently, if $\rho_i = \operatorname{cor}(X_i, F)$ and $\rho_{ij} = \operatorname{cor}(X_i, X_j)$, then

$$\rho_{ij} = \rho_i \rho_j.$$

A one-factor model does not allow arbitrary covariances. It imposes strong algebraic restrictions on the observed dependence pattern.

Economic reading

The assets comove because they all load on the same hidden market condition F.

Why this matters

A latent-factor story is not just a verbal explanation. It leaves a concrete statistical fingerprint: cross-covariances must line up in a very special low-dimensional pattern.

How can we detect that such a hidden factor may be present?

One-factor models impose testable restrictions

One-factor correlation pattern

The observed correlation matrix must have the form

$$\rho_{ij} = \rho_i \rho_j \qquad \text{for all } i \neq j.$$

Testable implications

If $1 \leq i < j < k \leq m$, then $\rho_{ij}\rho_{ik}\rho_{jk} = \rho_i^2 \rho_j^2 \rho_k^2 \geq 0.$

So under a one-factor model, the product of three pairwise correlations cannot be negative.

For any $1 \leq i < j < k < l \leq m$, we also have the **tetrad constraints**

$$\rho_{ij}\rho_{kl} = \rho_{ik}\rho_{jl} = \rho_{il}\rho_{jk}.$$

For classical testable implications of latent-variable models, see Spirtes, Glymour, and Scheines, *Causation, Prediction, and Search*, and Drton, Sturmfels, and Sullivant, *Lectures on Algebraic Statistics*.

A tiny one-factor check in R

Population-level relation: $\rho_i^2 = \frac{\rho_{ij}\rho_{ik}}{\rho_{jk}} \in [0, 1]$

```r
R <- matrix(c(1, 0.6, 0.3,
              0.6, 1, 0.5,
              0.3, 0.5, 1), 3, 3)

rho12 <- R[1,2]
rho13 <- R[1,3]
rho23 <- R[2,3]

rho12 * rho13 * rho23
```

```r
# implied squared correlations
rho2 <- c(rho12*rho13/rho23,
          rho12*rho23/rho13,
          rho13*rho23/rho12)

rho2

# all should lie in [0,1]
all(rho2 >= 0)
all(rho2 <= 1)
```

Interpretation

A negative product $\rho_{12}\rho_{13}\rho_{23}$ rules out a one-factor explanation. If the implied values also lie in $[0, 1]$, then a one-factor explanation is at least algebraically possible.

Parallel analysis: back to the general factor model

Let $\hat{R}$ be the **sample correlation matrix**, and let $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_m$ be its eigenvalues.

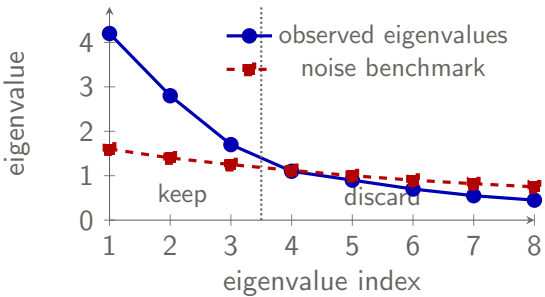
What do we expect under no factor structure? If the variables contain only idiosyncratic noise, then the eigenvalues of $\hat{R}$ should look like those from random data with the same sample size n and dimension m.

Parallel analysis

1. compute the eigenvalues of the observed sample correlation matrix,
2. simulate many noise datasets with the same n and m,
3. compute their sample-correlation eigenvalues,
4. use these as a benchmark,
5. keep only the factors whose observed eigenvalues exceed the benchmark.

For parallel analysis, see Horn (1965), *Psychometrika*.

Parallel analysis: visual intuition



Reading the plot. The blue curve comes from the observed sample correlation matrix. The red dashed curve is the typical eigenvalue pattern under pure noise.

Decision

Keep factors only while the observed eigenvalues remain above the noise benchmark. Here - **three** factors.

So the question is not whether an eigenvalue is large by itself, but whether it is larger than what noise alone would typically produce.

Choosing the number of factors in R

```r
set.seed(1)
library(MASS)
library(psych)

Lambda <- matrix(c(0.8, 0.2,
                   0.7, 0.1,
                   0.6, 0.3,
                   0.1, 0.7,
                   0.2, 0.8), 5, 2,
                 byrow = TRUE)
Psi <- diag(c(0.3,0.4,0.5,0.3,0.4))
Sigma <- Lambda %*% t(Lambda) + Psi
X <- mvrnorm(500, rep(0,5), Sigma)

fa.parallel(X, fa = "fa")
```

Interpretation

The plot compares the observed eigenvalues to a noise benchmark and suggests how many factors are worth retaining.

Why rotation matters

Recall: $X = \mu + WZ + \varepsilon, \qquad Z \sim N_r(0, I_r), \qquad \varepsilon \sim N_m(0, \Psi), \qquad Z \perp\!\!\!\perp \varepsilon.$

The factor model is not uniquely parameterized. If U is any orthogonal matrix, then

$$WZ = (WU)(U^\top Z),$$

and since $U^\top Z \sim N_r(0, I_r)$, the covariance matrix $\Sigma = WW^\top + \Psi$ does not change.

Why this matters Two loading matrices can define exactly the same fitted model, but one may be much easier to interpret than the other.

Varimax

Varimax chooses an orthogonal matrix U to rotate W into WU so as to maximize the variation of the squared loadings within each factor. Equivalently, it prefers columns whose entries are mostly either large or close to zero.

For rotation methods, see Kaiser (1958), *Psychometrika*.

Fitting and rotating a factor model in R

Unrotated fit

```
fit0 <- factanal(X, factors = 2,
                 rotation = "none")
fit0$loadings
```

Varimax rotation

```
fit1 <- factanal(X, factors = 2,
                 rotation = "varimax")
fit1$loadings
```

Interpretation

Varimax does not change the fitted covariance matrix. It only rotates the latent factor axes to make the loadings easier to read.

In a good outcome, each row has one or two clearly dominant entries, rather than many medium-sized loadings.

Application: personality-trait data

We analyze the `bfi` dataset from the `psych` package.

Dataset

- about 2,800 responses,
- 25 personality-related questions,
- items designed to reflect the Big Five personality traits: Neuroticism, Extraversion, Conscientiousness, Agreeableness, and Openness.

Responses are on a 1–6 agreement scale.

We discard demographic variables such as gender, age, and education.

For the `psych` package and the `bfi` data, see Revelle, *psych: Procedures for Psychological, Psychometric, and Personality Research*.

Factor analysis results

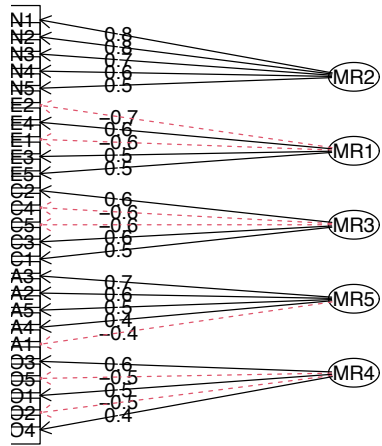
```r
library(psych)
data(bfi)

X <- na.omit(bfi[, 1:25])

fa.parallel(X, fa = "fa")

fit <- fa(X, nfactors = 5,
          rotate = "varimax")

fa.diagram(fit)
```



Empirical message

In this dataset, the factor structure is consistent with the Big Five interpretation.

Varimax rotation then makes the loading pattern easier to interpret.

What is the latent factor given the data?

Recall $X = \mu + WZ + \varepsilon$, $Z \sim N_r(0, I_r)$, $\varepsilon \sim N_m(0, \Psi)$.
The conditional distribution $Z \mid X$ is Gaussian: $Z|X = x \sim N_r(m(x), V)$, where

$$V = (I_r + W^\top \Psi^{-1} W)^{-1}, \qquad m(x) = V\, W^\top \Psi^{-1}(x - \mu).$$

Interpretation
The posterior mean $m(x) = \mathbb{E}[Z \mid X = x]$ is the best estimate of the hidden factor given the observed variables, while $V = \mathrm{var}(Z \mid X = x)$ measures the remaining uncertainty.

So in the Gaussian case, extracting the latent factor means computing the full conditional distribution of Z given X, not only a point estimate.

These formulas follow from the standard conditional formulas for multivariate Gaussians. See, for example, Bishop, *Pattern Recognition and Machine Learning*, Chapter 12.

How EM uses these conditional moments

Suppose we observe $x_1, \ldots, x_n$ but the factors $z_1, \ldots, z_n$ are hidden.

E-step Using the current parameters, compute for each observation

$$m_i = \mathbb{E}[Z_i \mid X_i = x_i], \qquad V_i = \mathrm{var}(Z_i \mid X_i = x_i).$$

Then replace the missing latent sufficient statistics by their conditional expectations:

$$\mathbb{E}[Z_i \mid X_i = x_i] = m_i, \qquad \mathbb{E}[Z_i Z_i^\top \mid X_i = x_i] = V_i + m_i m_i^\top.$$

Thus we obtain the expected cross-moments of the unobserved sample (X, Z).

M-step Update μ, W, and Ψ as if these expected sufficient statistics had been observed.

Interpretation

EM alternates between reconstructing the hidden part of the sample through conditional expectations and then refitting the model from these reconstructed moments.

For EM in latent-variable models, see Dempster, Laird, and Rubin (1977), *JRSS-B*.

Part 2: Latent tree models

Why latent tree models?

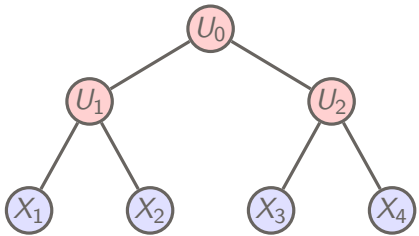
Sometimes one global factor is too crude, but a fully general latent structure is too flexible.

Latent tree idea

Observed variables sit at the leaves, and hidden variables occupy internal nodes. Nearby leaves share more hidden ancestry than distant leaves.

Economic reading

This is natural for hierarchical structure: firms inside sectors, sectors inside the macroeconomy, or regional demand inside a national cycle.



Interpretation

X_1 and X_2 are similar because they share a local hidden parent U_1. At a higher level, the two groups are connected through the more global hidden variable U_0.

Latent tree models as marginals of tree models

Start with a probabilistic model on a tree whose nodes include both observed and hidden variables. A **latent tree model** is what we get after marginalizing out the hidden nodes.

Special cases This includes familiar examples such as

- latent class / naive Bayes models,
- hidden Markov models,
- hierarchical Gaussian models.

Main point

Even if the hidden variables are not observed, the tree still imposes strong structure on the observed joint distribution. In the Gaussian case this generalizes the tetrad constraints in the one-factor case.

Gaussian latent tree models

Now suppose all variables in the tree are jointly Gaussian and centered.

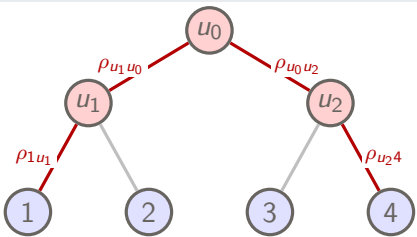
Edge correlations. For each edge (u, v) in the tree, write

$$\rho_{uv} = \mathrm{cor}(X_u, X_v).$$

Example

In the picture below,

$$\mathrm{cor}(X_1, X_4) = \rho_{1u_1}\rho_{u_1u_0}\rho_{u_0u_2}\rho_{u_24}.$$



Path-product formula

For any two nodes i and j in a Gaussian tree,

$$\mathrm{cor}(X_i, X_j) = \prod_{(u,v)\in\overline{ij}} \rho_{uv},$$

where $\overline{ij}$ is the unique path from i to j.

For Gaussian latent trees, see Zwiernik, *Latent tree models*, in the *Handbook of Graphical Models*.

From correlations to tree metrics

Recall: $\mathrm{cor}(X_i, X_j) = \prod_{(u,v)\in\overline{ij}} \rho_{uv}$ and $|\rho_{uv}| \in (0,1)$.

Because correlations multiply along paths, logarithms turn them into sums:

$$-\log|\mathrm{cor}(X_i, X_j)| = \sum_{(u,v)\in\overline{ij}} -\log|\rho_{uv}| > 0.$$

Tree metric

This means the quantities

$$d(i,j) := -\log|\mathrm{cor}(X_i, X_j)|$$

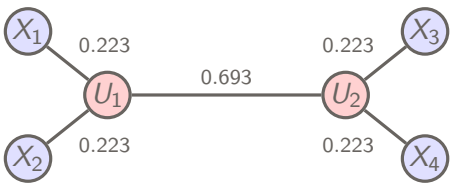
behave like path lengths on a tree.

This is why pairwise correlations can be used to reconstruct a hidden tree.

A concrete Gaussian latent tree example

Observed correlation matrix

$$R = \begin{pmatrix} 1 & 0.64 & 0.32 & 0.32 \\ 0.64 & 1 & 0.32 & 0.32 \\ 0.32 & 0.32 & 1 & 0.64 \\ 0.32 & 0.32 & 0.64 & 1 \end{pmatrix}$$



This matrix comes from a latent tree in which the four outer edges have correlation 0.8 and the middle edge has correlation 0.5. For example,

$$\mathrm{cor}(X_1, X_2) = 0.8^2 = 0.64,$$
$$\mathrm{cor}(X_1, X_3) = 0.8 \cdot 0.5 \cdot 0.8 = 0.32.$$

The edge labels are the tree distances

$$-\log(0.8) \approx 0.223, \quad -\log(0.5) \approx 0.693.$$

Applications beyond economics

Latent trees are useful when hidden structure is *hierarchical* rather than global.

Psychometrics
Broad latent traits may split into narrower dimensions, with observed items at the leaves.

Hierarchical topic detection
Observed binary word variables sit at the leaves, while hidden nodes capture word co-occurrence patterns at different levels of specificity.

Phylogenetics
Observed species sit at the leaves, hidden ancestral species lie at internal nodes, and genetic distances behave approximately like path lengths on a tree.

All three examples share the same mathematical pattern: observed leaves, hidden internal nodes, and additive path structure.

Pointers: Reise (2012), *Multivariate Behavioral Research*; Liu, Zhang, and Chen (2014), *Machine Learning*; Semple and Steel, *Phylogenetics*.

Relation to hierarchical clustering

Once correlations are transformed into distances, $d(i,j) = -\log|\operatorname{cor}(X_i, X_j)|$, we obtain an $m \times m$ **dissimilarity matrix** between the observed variables $X_1, \ldots, X_m$.

Hierarchical clustering takes such a dissimilarity matrix and builds a nested sequence of groups of variables, usually displayed as a **dendrogram**.

But there is a difference

Hierarchical clustering always returns a dendrogram, even if the dissimilarities do not truly come from a tree metric. Latent-tree reconstruction asks a stronger question: do these distances really behave like sums along tree paths?

So hierarchical clustering is mainly an exploratory summary of similarity among variables, while latent-tree models add a probabilistic interpretation and a stronger structural claim.

How hierarchical clustering works

Suppose we are given a dissimilarity matrix $D = (d(i,j))_{1 \le i,j \le m}$ for the variables $X_1, \ldots, X_m$.

Basic algorithm

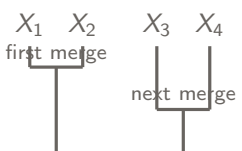
1. Start with each variable in its own cluster.
2. Find the two clusters that are closest.
3. Merge them into one larger cluster.
4. Update the distance between this new cluster and the remaining ones.
5. Repeat until everything is merged.

The output is a **dendrogram**: a nested sequence of groups formed by successive merges.

Different linkage rules

The update in step 4 depends on the chosen rule:

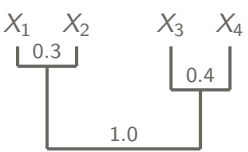
- **single linkage:** minimum distance,
- **complete linkage:** max distance,
- **average linkage:** average distance.



Hierarchical clustering: a tiny example

Dissimilarity matrix

$$D = \begin{pmatrix} 0 & 0.3 & 1.2 & 1.4 \\ 0.3 & 0 & 1.0 & 1.3 \\ 1.2 & 1.0 & 0 & 0.4 \\ 1.4 & 1.3 & 0.4 & 0 \end{pmatrix}.$$



Step 1 The smallest dissimilarity is $d(1,2) = 0.3$, so we first merge X_1 and X_2. Now we have $\{1,2\}, \{3\}, \{4\}$.

Updated distances (with single linkage):

$d(\{1,2\},3) = \min\{d_{13}, d_{23}\} = 1,\ d(\{1,2\},4) = \min\{d_{14}, d_{24}\} = 1.3$

Step 2 The smallest dissimilarity is $d(3,4) = 0.4$, so we next merge X_3 and X_4. Now we have $\{1,2\}, \{3,4\}$.

Updated distances (with single linkage):

$d(\{1,2\},\{3,4\}) = \min\{d_{13}, d_{23}, d_{14}, d_{24}\} = 1$

A stock-market example: exploratory hierarchy from correlations

The dataset `stockdata` contains stock prices together with sector information. We compute log-returns, transform correlations into distances, and then apply single-linkage clustering.

```
library(huge)
data(stockdata)
set.seed(1)
ns <- 80
sample.stocks <- sample(1:ncol(stockdata$data), ns)

dat <- log(
  stockdata$data[2:1258, sample.stocks] /
  stockdata$data[1:1257, sample.stocks]
)
dat <- scale(dat)

R <- cor(dat)
D <- -log(abs(R))
hc <- hclust(as.dist(D), method = "single")
```

Interpretation

Stocks with strong correlations have smaller distances $d(i,j) = -\log|R_{ij}|$. If the market has hidden hierarchical structure, then firms from the same sector should tend to appear close together.

This is an exploratory construction, not a full latent-tree fit. But it gives a first data-driven picture of possible hidden hierarchy.

Visualizing the stock hierarchy

We can display the result as a dendrogram.

```
labs <- stockdata$info[sample.stocks, 1]

plot(hc, labels = labs,
     cex = 0.55,
     main = "")

# optional: color labels by sector
sector <- as.factor(stockdata$info[sample.stocks, 2])
lab.col <- as.numeric(sector)

plot(hc, labels = FALSE, cex = 0.55, main = "")
text(x = 1:length(hc$order), y = par("usr")[3] - 0.02,
     labels = labs[hc$order], srt = 90, adj = 1,
     xpd = TRUE, col = lab.col[hc$order], cex = 0.55)
```

What to look for

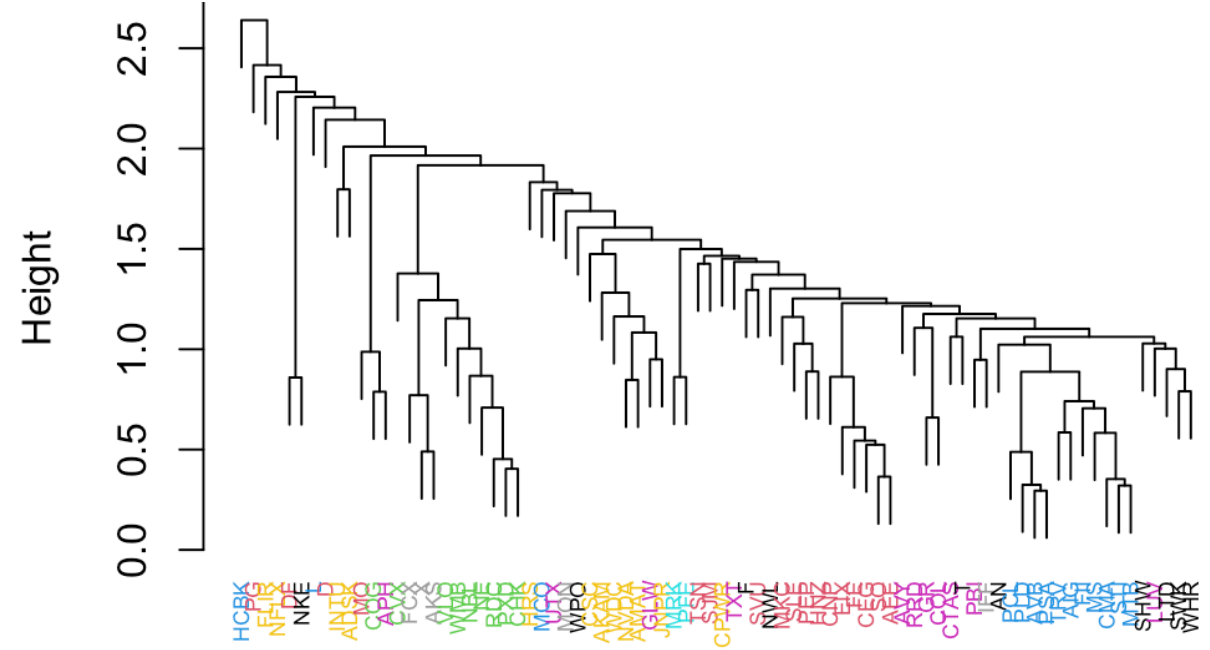
If the construction is meaningful, one may see

- firms from the same sector merging early,
- related sectors joining at intermediate heights.

Coloring labels by sector helps check whether the recovered hierarchy lines up with economic structure already known from the data.

Something to think before the last lecture

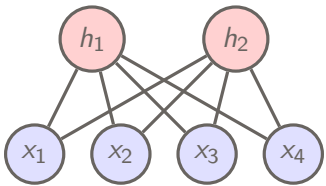
Because we use $d(i,j) = -\log|R_{ij}|$, strongly negative and strongly positive correlations are both treated as close. Whether this is sensible depends on the application.


Height
0.0
0.5
1.0
1.5
2.0
2.5

Part 3: Restricted Boltzmann machines

Restricted Boltzmann machines

A restricted Boltzmann machine (RBM) is a **bipartite undirected latent-variable model** with visible units x and hidden units h.



Restriction

There are no visible-visible edges and no hidden-hidden edges. So the model is simple *conditionally*, even though the marginal dependence structure can be complicated.

For RBMs, see Hinton (2002), *Neural Computation*, and Fischer and Igel (2012), *Pattern Recognition*.

RBMs versus factor models

Both factor models and RBMs introduce a hidden layer, but the simplification happens in different places.

Factor model Usually written as

$$X = \mu + WZ + \varepsilon.$$

The latent variable Z is simple *marginally*, often Gaussian with independent coordinates. Dependence among the observed variables is explained through hidden factors.

Restricted Boltzmann Machine
An undirected bipartite model with joint law

$$P(x, h) \propto \exp\big(a^\top x + b^\top h + x^\top W h\big).$$

The hidden units are simple *conditionally*: once x is fixed, the coordinates of h become independent.

A factor model gives a mostly linear latent representation. An RBM already gives a **nonlinear** latent representation through conditional activation probabilities.

Why the hidden layer factorizes conditionally

Fix the visible vector x. Then

$$P(h \mid x) \propto \exp\left(b^\top h + x^\top W h\right).$$

Since

$$b^\top h + x^\top W h = \sum_{j=1}^{q} h_j \left(b_j + w_j^\top x\right),$$

we obtain

$$P(h \mid x) \propto \prod_{j=1}^{q} \exp\Big(h_j (b_j + w_j^\top x)\Big).$$

Conditional independence

Therefore,

$$P(h \mid x) = \prod_{j=1}^{q} P(h_j \mid x).$$

Likewise,

$$P(x \mid h) = \prod_{i=1}^{m} P(x_i \mid h).$$

So the RBM is complicated marginally, but very simple once one layer is conditioned on. This conditional factorization is the key computational feature of the model.

From conditional factorization to nonlinear features

If the hidden units are binary, then each conditional probability has logistic form:

$$P(h_j = 1 \mid x) = \frac{\exp(b_j + w_j^\top x)}{1 + \exp(b_j + w_j^\top x)} = \sigma(b_j + w_j^\top x), \qquad \sigma(t) = \frac{1}{1 + e^{-t}}.$$

Factor model

A feature is often the posterior mean

$$\mathbb{E}[Z \mid X = x],$$

which is a linear-Gaussian latent score.

RBM

A feature is the activation vector

$$\phi(x) = \big(P(h_1 = 1 \mid x), \ldots, P(h_q = 1 \mid x)\big).$$

This is a nonlinear representation of the input.

Bridge to neural nets

To compute $\phi(x)$ map $x \mapsto b + W^\top x$ and apply the activation function σ coordinatewise.

RBMs form a bridge from latent-variable models to neural-network architectures.

RBM feature extraction in R

Assume the weight matrix W and hidden intercepts b have already been learned.

```r
set.seed(1)
X <- matrix(rbinom(500*4, 1, 0.5),
            nrow = 500, ncol = 4)
W <- matrix(c(1.2, -0.8,
              0.5, 1.1,
             -0.7, 0.9,
              1.0, 0.6),
            nrow = 4, ncol = 2,byrow = TRUE)
b <- c(0.3, -0.2)
```

```r
# linear scores for the hidden units
eta <- X %*% W
# add the hidden intercepts
eta[,1] <- eta[,1] + b[1]
eta[,2] <- eta[,2] + b[2]
# convert scores into probabilities
Hprob <- 1 / (1 + exp(-eta))

head(Hprob)
```

For each observation x, the row of `Hprob` gives $\phi(x) = \big(P(h_1 = 1 \mid x), P(h_2 = 1 \mid x)\big)$.

So the original binary input is replaced by a learned low-dimensional feature vector.

Interpretation

This is the basic feature-extraction step in an RBM: raw observed variables are turned into latent feature scores summarizing hidden patterns in the data.

Take-away

- factor analysis explains dependence through a few hidden common shocks,
- latent tree models explain dependence through hierarchical hidden structure,
- RBMs show how latent-variable ideas connect naturally to neural-network representations.

Different latent models explain dependence in different ways: global factors, hidden hierarchies, or distributed nonlinear features.

Looking ahead

Next lecture
We will return to hidden variables from a more causal perspective: unobserved confounding, proxy controls, and what survives when the clean DAG picture breaks.

Bridge forward

Today the latent variables explained dependence. Next time they will also threaten identification.