Ch2 Lecture 4

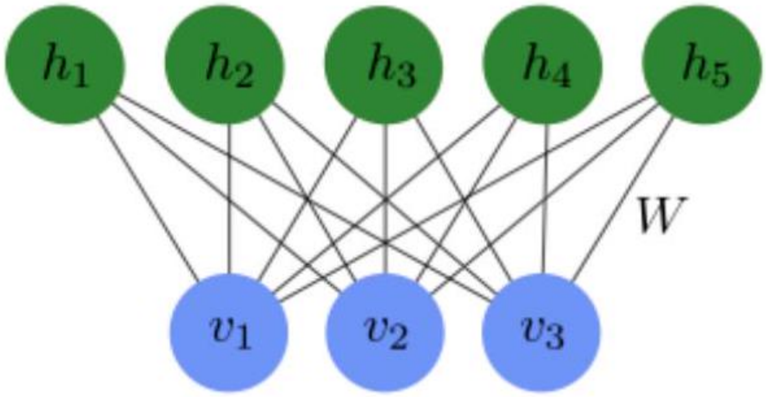
MCMC

Idea of MCMC

Restricted Boltzmann Machine

Intro to the idea of a Restricted Boltzmann Machine

Math of the RBM



From https://ml-lectures.org/docs/unsupervised_learning/ml_unsuperv
1.html

. . .

States are determined by an **energy function** $E(\mathbf{v}, \mathbf{h})$.

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{ij} v_i W_{ij} h_j$$

. . .

Then the probability distribution is given by the Boltzmann distribution:

$P_{\mathrm{rbm}}(\mathbf{v},\mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})}$ where $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$

The probability of a visible vector $\mathbf{v}$ is given by marginalizing over the hidden variables:

$$P_{\mathrm{rbm}}(\mathbf{v}) = \sum_{\mathbf{h}} P_{\mathrm{rbm}}(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\sum_{h} e^{-E(\mathbf{v},\mathbf{h})}$$

Conveniently, this gives each visible unit an **independent** probability of activation:

$$P_{\mathrm{rbm}}\left(v_i = 1|\mathbf{h}\right) = \sigma\left(a_i + \sum_j W_{ij}h_j\right), \quad i = 1, \dots, n_{\mathrm{v}}$$

The same is true for hidden units, given the visible units:

$$P_{\mathrm{rbm}}\left(h_j = 1|\mathbf{v}\right) = \sigma\left(b_j + \sum_i v_i W_{ij}\right) \quad j = 1, \dots, n_{\mathrm{h}}$$

Training

Consider a set of binary input data $\mathbf{x}_k, k = 1, \dots, M$, drawn from a probability distribution $P_{\mathrm{data}}\left(\mathbf{x}\right)$.

Goal: tune the parameters $\{\mathbf{a}, \mathbf{b}, W\}$ such that after training $P_{\mathrm{rbm}}(\mathbf{x}) \approx P_{\mathrm{data}}(\mathbf{x})$.

. . .

To do this, we need to be able to estimate P_{rbm}!

Unfortunately, this is often intractable, because it requires calculating the partition function Z.

Details of the training

We want to maximize the log-likelihood of the data under the model:

$$L(\mathbf{a}, \mathbf{b}, W) = -\sum_{k=1}^{M} \log P_{\text{rbm}}\left(\mathbf{x}_k\right)$$

. . .

Take derivatives of this with respect to the parameters, and use gradient descent:

$$\frac{\partial L(\mathbf{a}, \mathbf{b}, W)}{\partial W_{ij}} = -\sum_{k=1}^{M} \frac{\partial \log P_{\text{rbm}}\left(\mathbf{x}_k\right)}{\partial W_{ij}}$$

The derivative has two terms:

$$\frac{\partial \log P_{\text{rbm}}(\mathbf{x})}{\partial W_{ij}} = x_i P_{\text{rbm}}\left(h_j = 1|\mathbf{x}\right) - \sum_{\mathbf{v}} v_i P_{\text{rbm}}\left(h_j = 1|\mathbf{v}\right) P_{\text{rbm}}(\mathbf{v})$$

. . .

Use this to update the weights:

$$W_{ij} \rightarrow W_{ij} - \eta \frac{\partial L(a, b, W)}{\partial W_{ij}}$$

Problem: the second term in the derivative is intractable! It has $2^{n_{\text{v}}}$ terms:

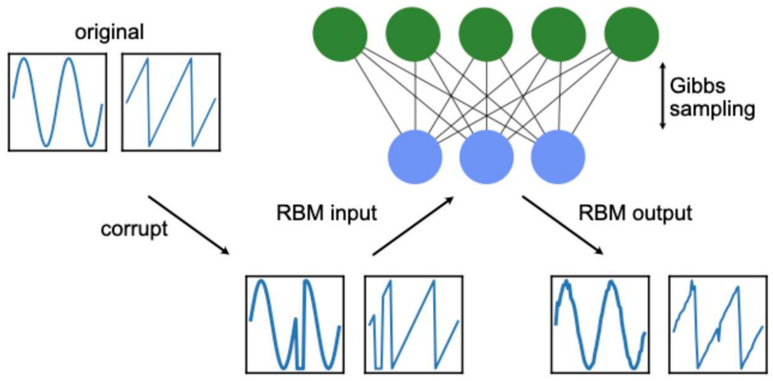
$$\sum_{\mathbf{v}} v_i P_{\text{rbm}}\left(h_j = 1|\mathbf{v}\right) P_{\text{rbm}}(\mathbf{v})$$

Instead, we will use **Gibbs sampling** to estimate $P_{\text{rbm}}(\mathbf{v})$.

Gibbs sampling to the rescue

```
Input: Any visible vector $\mathbf{v}(0)$
Output: Visible vector $\mathbf{v}(r)$
for: $n = 1\backslash$ dots $r$
sample $\mathbf{h}(n)$ from $P_{\text{rbm}}(\mathbf{h}\mathbf{v} = \mathbf{v}(n-1))$
sample $\mathbf{v}(n)$ from $P_{\text{rbm}}(\mathbf{v}\mathbf{h} = \mathbf{h}(n))$ end
```

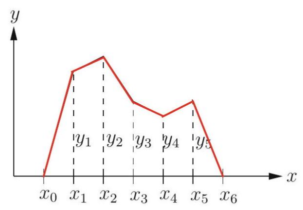
Using an RBM



LU Factorization

Suppose we want to solve a nonsingular linear system $Ax = b$ repeatedly, with different choices of b.

. . .



$$-y_{i-1} + 2y_i - y_{i+1} = \frac{h^2}{K} f(x_i), i = 1, 2, \dots, n$$

. . .

Perhaps you want to experiment with different functions for the heat source term.

. . .

What do we do? Each time, we create the augmented matrix $\widetilde{A} = [A \mid b]$, then get it into reduced row echelon form.

. . .

Each time change b, we have to redo all the work of Gaussian or Gauss-Jordan elimination !

. . .

Especially frustrating because the main part of our work is the same: putting the part of $\widetilde{A}$ corresponding to the coefficient matrix A into reduced row echelon form.

LU Factorization: Saving that work

Goal: Find a way to record our work on A, so that solving a new system involves very little additional work.

LU Factorization: Let A be an $n \times n$ matrix. An LU factorization of A is a pair of $n \times n$ matrices L, U such that

1. L is lower triangular.
2. U is upper triangular.
3. $A = LU$.

. . .

Why is this so wonderful? Triangular systems $A\mathbf{x} = \mathbf{b}$ are easy to solve.

Remember: If A is upper triangular, we can solve for the last variable, then the next-to-last variable, etc.

Solving an upper triangular system

Let's say we have the following system:

$Ax = b$ where A is the upper-triangular matrix $A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}$, and we want to solve for $b = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$.

. . .

We form the augmented matrix $\widetilde{A} = [A|b] = \begin{bmatrix} 2 & 1 & 0 & | & 1 \\ 0 & 1 & -1 & | & 1 \\ 0 & 0 & -1 & | & -2 \end{bmatrix}$.

. . .

Back substitution:

1. Last equation: $-x_3 = -2$, so $x_3 = 2$.
2. Substitute this value into the second equation, $x_2 - x_3 = 1$, so $x_2 = 3$.
3. Finally, we substitute x_2 and x_3 into the first equation, $2x_1 + x_2 = 1$, so $x_1 = -1$.

Solving a lower triangular system

If A is lower triangular, we can solve for the first variable, then the second variable, etc.

Let's say we have the following system:

$Ay = b$ where A is the lower-triangular matrix $A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$, and we want to solve for $b = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

. . .

We form the augmented matrix $\widetilde{A} = [A|b] = \begin{bmatrix} 1 & 0 & 0 & | & 1 \\ -1 & 1 & 0 & | & 0 \\ 1 & 2 & 1 & | & 1 \end{bmatrix}$.

. . .

Forward substitution:

1. First equation: $y_1 = 1$.

2. Substitute this value into the second equation, $-y_1+y_2 = 0$, so $y_2 = 1$.
3. Finally, we substitute y_1 and y_2 into the third equation, $y_1 + 2y_2 + y_3 = 1$, so $y_3 = -2$.

. . .

This was just as easy as solving the upper triangular system!

Solving $Ax = b$ with LU factorization

Now suppose we want to solve $Ax = b$ and we know that $A = LU$. The original system becomes $LUx = b$.

Introduce an intermediate variable $y = Ux$. Our system is now $Ly = b$. Now perform these steps:

1. **Forward solve**: Solve lower triangular system $Ly = b$ for the variable y.
2. **Back solve**: Solve upper triangular system $Ux = y$ for the variable x.
3. This does it!

. . .

Once we have the matrices L, U, the right-hand sides only come when solving the two triangular systems. Easy!

Example

You are given that

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 0 & -1 \\ 2 & 3 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}.$$

Solve this system for $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

. . .

Forward solve:

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$y_1 = 1$, then $y_2 = 0 + 1y_1 = 1$, then $y_3 = 1 - 1y_1 - 2y_2 = -2$.

Back solve:

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$$

$x_3 = -2/(-1) = 2$, then $x_2 = 1 + x_3 = 3$, then $x_1 = (1 - 1x_2)/2 = -1$.

When we can do LU factorization

- Not all square matrices have LU factorizations! This one doesn't: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- If Gaussian elimination can be performed on the matrix A **without row exchanges**, then the factorization exists
    - (it's really a by-product of Gaussian elimination.)
- If row exchanges are needed, there is still a factorization that will work, but it's a bit more complicated.

Intuition behind LU factorization

Example

Here we do Gaussian elimination on the matrix $A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 0 & -1 \\ 2 & 3 & -3 \end{bmatrix}$:

$$\begin{bmatrix} 2 & 1 & 0 \\ -2 & 0 & -1 \\ 2 & 3 & -3 \end{bmatrix} \xrightarrow[E_{31}(-1)]{E_{21}(1)} \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 2 & -3 \end{bmatrix} \xrightarrow[E_{32}(-2)]{\longrightarrow} \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}$$

Let's put those elementary row operations into matrix form. There were three of them:

1. $E_{21}(1)$: $\begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
2. $E_{31}(-1)$: $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -1 & 0 & 1\end{bmatrix}$
3. $E_{32}(-2)$: $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & -2 & 1\end{bmatrix}$

The **inverses** of these matrices are

1. $\begin{bmatrix}1 & 0 & 0\\ -1 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$, $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 1 & 0 & 1\end{bmatrix}$, and $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 2 & 1\end{bmatrix}$.

The product of all these matrices is

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & -2 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ -1 & -2 & 1\end{bmatrix}$$

This is a lower triangular matrix, and it is the inverse of the matrix we used to do Gaussian elimination.

We can also see that the entries below the diagonal are the negatives of the multipliers we used in the elimination steps.

How to Compute an LU Factorization (no row swaps)

Algorithm:

1. Start with $U := A$ and $L := I$ (identity matrix).
2. For each pivot column $k = 1, \ldots, n-1$:
    - For each row $i = k+1, \ldots, n$:

   - Compute the multiplier: $\ell_{ik} := U_{ik}/U_{kk}$
   - Update the row in U: subtract ℓ_{ik} times row k from row i
   - Store the multiplier in L: set $L_{ik} := \ell_{ik}$

3. At the end, U is upper triangular, L is unit lower triangular, and $A = LU$.

Note: If a pivot U_{kk} is zero (or you decide to swap rows), you need a permutation matrix: $PA = LU$ (see PLU factorization section below).

This is exactly what "store the multipliers as you go" means.

Steps to LU factorization

Let $\left[a_{ij}^{(k)}\right]$ be the matrix obtained from A after using the k th pivot to clear out entries below it.

. . .

(The original matrix is $A = \left[a_{ij}^{(0)}\right]$)

. . .

All the row operations we will use include ratios $\left(-a_{ij}/a_{jj}\right)$.

. . .

The row-adding elementary operations are of the form

$E_{ij}\left(-a_{ij}^{(k)}/a_{jj}^{(k)}\right)$

. . .

We can give these ratios a name: **multipliers**.

$m_{ij} = -a_{ij}^{(k)}/a_{jj}^{(k)}$, where $i > j$

. . .

If Gaussian elimination is used without row exchanges on the nonsingular matrix A, resulting in the upper triangular matrix U, and if L is the unit lower triangular matrix whose entries below the diagonal are the negatives of the multipliers m_{ij}, then $A = LU$.

Storing the multipliers as we go

For efficiency, we can just "store" the multipliers in the lower triangular part of the matrix on the left as we go along, since that will be zero anyways.

$$\begin{bmatrix} (2) & 1 & 0 \\ -2 & 0 & -1 \\ 2 & 3 & -3 \end{bmatrix} \xrightarrow[E_{31}(-1)]{E_{21}(1)} \begin{bmatrix} 2 & 1 & 0 \\ -1 & (1) & -1 \\ 1 & 2 & -3 \end{bmatrix} \xrightarrow[E_{32}(-2)]{\longrightarrow} \begin{bmatrix} 2 & 1 & 0 \\ -1 & 1 & -1 \\ 1 & 2 & -1 \end{bmatrix}.$$

. . .

Now we read off the results from the final matrix:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & 2 & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}$$

Superaugmented matrix

Could we just keep track by using the superaugmented matrix, like we did last lecture? What would that look like?

pause

$$\left[\begin{array}{cccccc} 2 & 1 & 0 & 1 & 0 & 0 \\ -2 & 0 & -1 & 0 & 1 & 0 \\ 2 & 3 & -3 & 0 & 0 & 1 \end{array}\right] \xrightarrow[E_{31}(-1)]{E_{21}(1)} \left[\begin{array}{cccccc} 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & 1 & 0 \\ 0 & 2 & -3 & -1 & 0 & 1 \end{array}\right] \xrightarrow[E_{32}(-2)]{\longrightarrow} \left[\begin{array}{cccccc} 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & -3 & -2 & 1 \end{array}\right]$$

. . .

Our superaugmented matrix does become an upper triangular matrix on the left and a lower triangular matrix on the right.

Unfortunately, the lower triangular matrix on the right is $\tilde{L}^{-1}$, not $\tilde{L}$.

So we can't just read off L and U from the superaugmented matrix.

PLU factorization

What if we need row exchanges?

- We could start off by doing all the row-exchanging elementary operations that we need, and store the product of these row-exchanging matrices as a matrix P.
- This product is called a **permutation matrix**
- Applying the correct permuatation matrix P to A, we get a matrix for which Gaussian elimination will succeed without further row exchanges.

. . .

Now we have a theorem that applies to all nonsingular matrices:

If A is a nonsingular matrix, then there exists a permutation matrix P, upper triangular matrix U, and unit lower triangular matrix L such that $PA = LU$.

. . .

So, if you've got a nonsingular matrix A, you can always find a permutation matrix P, an upper triangular matrix U, and a unit lower triangular matrix L that satisfy $PA = LU$. Pretty neat, huh?