CSCI567 Machine Learning (Fall 2025)

Haipeng Luo

University of Southern California

Oct 31, 2025

Administration

Reminder: HW3 is due on Nov 5th.

Outline

Outline

1. Principal Component Analysis (PCA)
2. Markov models
3. Hidden Markov Model

Dimensionality reduction

Dimensionality reduction is yet another important unsupervised learning problem.

Dimensionality reduction

Dimensionality reduction is yet another important unsupervised learning problem.

Goal: reduce the dimensionality of a dataset so

- it is easier to visualize and discover patterns

Dimensionality reduction

Dimensionality reduction is yet another important unsupervised learning problem.

Goal: reduce the dimensionality of a dataset so

- it is easier to visualize and discover patterns
- it takes less time and space to process for any applications (classification, regression, clustering, etc)

Dimensionality reduction

Dimensionality reduction is yet another important unsupervised learning problem.

Goal: reduce the dimensionality of a dataset so

- it is easier to visualize and discover patterns
- it takes less time and space to process for any applications (classification, regression, clustering, etc)
- noise is reduced
- $\cdots$

Dimensionality reduction

Dimensionality reduction is yet another important unsupervised learning problem.

Goal: reduce the dimensionality of a dataset so

- it is easier to visualize and discover patterns
- it takes less time and space to process for any applications (classification, regression, clustering, etc)
- noise is reduced
- $\cdots$

There are many approaches, we focus on a linear method:
Principal Component Analysis (PCA)

Example

picture from here

Consider the following dataset:

- 17 features, each represents the average consumption of some food

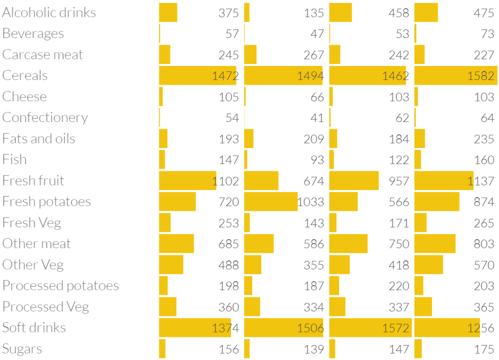
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566	874
Fresh Veg	253	143	171	265
Other meat	685	586	750	803
Other Veg	488	355	418	570
Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175

Example

picture from here

Consider the following dataset:

- 17 features, each represents the average consumption of some food
- 4 data points, each represents some country

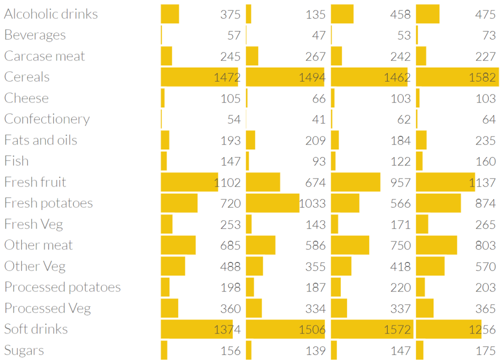


Example

picture from here

Consider the following dataset:

- 17 features, each represents the average consumption of some food
- 4 data points, each represents some country



What can you tell?

Example

picture from here

Consider the following dataset:

- 17 features, each represents the average consumption of some food
- 4 data points, each represents some country

Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566	874
Fresh Veg	253	143	171	265
Other meat	685	586	750	803
Other Veg	488	355	418	570
Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175

What can you tell?

Hard to say anything looking at all these 17 features.

Example

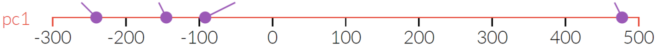
picture from here

PCA can help us!

Example

picture from here

PCA can help us! The first principal component of this dataset:

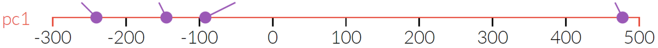


i.e. we reduce the dimensionality from 17 to just 1.

Example

picture from here

PCA can help us! The first principal component of this dataset:



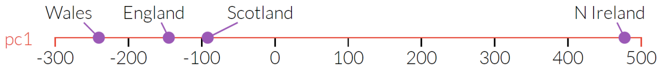
i.e. we reduce the dimensionality from 17 to just 1.

Now one data point is clearly different from the rest!

Example

picture from here

PCA can help us! The first principal component of this dataset:



i.e. we reduce the dimensionality from 17 to just 1.

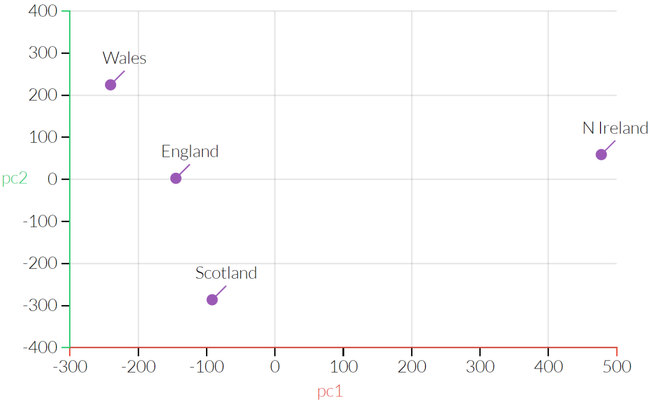
Now one data point is clearly different from the rest!

That turns out to be data from Northern Ireland, *the only country not on the island of Great Britain out of the 4 samples.*

Example

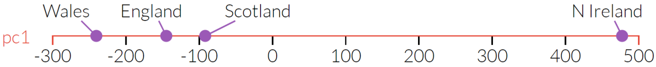
picture from here

PCA can find the **second (and more) principal component** of the data too:



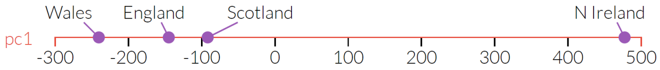
High level idea

How does PCA find these principal components (PC)?



High level idea

How does PCA find these principal components (PC)?



The first PC is in fact **the direction with the most variance**, i.e. the direction where the data is most spread out.

Finding the first PC

More formally, we want to find a direction $\boldsymbol{v} \in \mathbb{R}^{\mathsf{D}}$ with $\|\boldsymbol{v}\|_2 = 1$, so that the projection of the dataset on this direction has the most variance,

Finding the first PC

More formally, we want to find a direction $\boldsymbol{v} \in \mathbb{R}^{\mathsf{D}}$ with $\|\boldsymbol{v}\|_2 = 1$, so that the projection of the dataset on this direction has the most variance, i.e.

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left(\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v} - \frac{1}{N} \sum_{m} \boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{v} \right)^2$$

Finding the first PC

More formally, we want to find a direction $\boldsymbol{v} \in \mathbb{R}^{\mathsf{D}}$ with $\|\boldsymbol{v}\|_2 = 1$, so that the projection of the dataset on this direction has the most variance, i.e.

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left(\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v} - \frac{1}{N} \sum_{m} \boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{v} \right)^2$$

- $\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v}$ is exactly the projection of $\boldsymbol{x}_n$ onto the direction $\boldsymbol{v}$

Finding the first PC

More formally, we want to find a direction $\boldsymbol{v} \in \mathbb{R}^{\mathsf{D}}$ with $\|\boldsymbol{v}\|_2 = 1$, so that the projection of the dataset on this direction has the most variance, i.e.

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left(\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v} - \frac{1}{N} \sum_m \boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{v} \right)^2$$

- $\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v}$ is exactly the projection of $\boldsymbol{x}_n$ onto the direction $\boldsymbol{v}$
- if we pre-center the data, i.e. let $\boldsymbol{x}'_n = \boldsymbol{x}_n - \frac{1}{N} \sum_m \boldsymbol{x}_m$, then the objective simply becomes

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left({\boldsymbol{x}'_n}^{\mathrm{T}} \boldsymbol{v} \right)^2$$

Finding the first PC

More formally, we want to find a direction $\boldsymbol{v} \in \mathbb{R}^{\mathsf{D}}$ with $\|\boldsymbol{v}\|_2 = 1$, so that the projection of the dataset on this direction has the most variance, i.e.

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left(\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v} - \frac{1}{N} \sum_m \boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{v} \right)^2$$

- $\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v}$ is exactly the projection of $\boldsymbol{x}_n$ onto the direction $\boldsymbol{v}$
- if we pre-center the data, i.e. let $\boldsymbol{x}'_n = \boldsymbol{x}_n - \frac{1}{N} \sum_m \boldsymbol{x}_m$, then the objective simply becomes

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left({\boldsymbol{x}'_n}^{\mathrm{T}} \boldsymbol{v} \right)^2 = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{x}'_n {\boldsymbol{x}'_n}^{\mathrm{T}} \right) \boldsymbol{v}$$

Finding the first PC

More formally, we want to find a direction $\boldsymbol{v} \in \mathbb{R}^{\mathsf{D}}$ with $\|\boldsymbol{v}\|_2 = 1$, so that the projection of the dataset on this direction has the most variance, i.e.

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left(\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v} - \frac{1}{N} \sum_m \boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{v} \right)^2$$

- $\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{v}$ is exactly the projection of $\boldsymbol{x}_n$ onto the direction $\boldsymbol{v}$
- if we pre-center the data, i.e. let $\boldsymbol{x}_n' = \boldsymbol{x}_n - \frac{1}{N} \sum_m \boldsymbol{x}_m$, then the objective simply becomes

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{n=1}^{N} \left(\boldsymbol{x}_n'^{\mathrm{T}} \boldsymbol{v} \right)^2 = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{x}_n' \boldsymbol{x}_n'^{\mathrm{T}} \right) \boldsymbol{v}$$

- we will simply assume $\{\boldsymbol{x}_n\}$ is centered (to avoid notation $\boldsymbol{x}_n'$)

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N\times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right) \boldsymbol{v}$$

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U}\boldsymbol{v}$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U} \boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U} \boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^{\top} \boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Note: $\sum_{i=1}^{D} (\boldsymbol{u}_i^{\top} \boldsymbol{v})^2 = \boldsymbol{v}^{\top} \left(\sum_{i=1}^{D} \boldsymbol{u}_i \boldsymbol{u}_i^{\top} \right) \boldsymbol{v}$

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U}\boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^{\top}\boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Note: $\sum_{i=1}^{D} (\boldsymbol{u}_i^{\top}\boldsymbol{v})^2 = \boldsymbol{v}^{\top} \left(\sum_{i=1}^{D} \boldsymbol{u}_i\boldsymbol{u}_i^{\top}\right) \boldsymbol{v} = \boldsymbol{v}^{\top}\boldsymbol{v}$

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U}\boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^{\top}\boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Note: $\sum_{i=1}^{D} (\boldsymbol{u}_i^{\top}\boldsymbol{v})^2 = \boldsymbol{v}^{\top} \left(\sum_{i=1}^{D} \boldsymbol{u}_i\boldsymbol{u}_i^{\top}\right) \boldsymbol{v} = \boldsymbol{v}^{\top}\boldsymbol{v} = 1,$

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}\right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U} \boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Note: $\sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 = \boldsymbol{v}^\top \left(\sum_{i=1}^{D} \boldsymbol{u}_i \boldsymbol{u}_i^\top \right) \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{v} = 1,$ so the maximum above is λ_1,

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N \times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right) \boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U}\boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Note: $\sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 = \boldsymbol{v}^\top \left(\sum_{i=1}^{D} \boldsymbol{u}_i \boldsymbol{u}_i^\top\right) \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{v} = 1,$ so the maximum above is λ_1, realized by $\boldsymbol{v} = \boldsymbol{u}_1$.

Finding the first PC

With $\boldsymbol{X} \in \mathbb{R}^{N\times \mathsf{D}}$ being the data matrix (as in Lec 2), we want

$$\max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}}\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)\boldsymbol{v} = \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}} \end{bmatrix} \boldsymbol{U}\boldsymbol{v}$$

$$= \max_{\boldsymbol{v}:\|\boldsymbol{v}\|_2=1} \sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 \lambda_i,$$

where the columns of U^{T} are unit **eigenvectors** $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\mathsf{D}}$ of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}} \geq 0$.

Note: $\sum_{i=1}^{D} (\boldsymbol{u}_i^\top \boldsymbol{v})^2 = \boldsymbol{v}^\top \left(\sum_{i=1}^{D} \boldsymbol{u}_i \boldsymbol{u}_i^\top\right) \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{v} = 1,$ so the maximum above is λ_1, realized by $\boldsymbol{v} = \boldsymbol{u}_1$.

Conclusion: <u>the first PC is the top eigenvector of the covariance matrix!</u>

Finding the other PCs

If v_1 is the first PC, then the **second PC** is found via

$$\max_{\boldsymbol{v}_2 : \|\boldsymbol{v}_2\|_2 = 1, \boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_2 = 0} \boldsymbol{v}_2^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \right) \boldsymbol{v}_2$$

i.e. the direction that maximizes the variance among all other dimensions

Finding the other PCs

If v_1 is the first PC, then the **second PC** is found via

$$\max_{\boldsymbol{v}_2 : \|\boldsymbol{v}_2\|_2 = 1, \boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_2 = 0} \boldsymbol{v}_2^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \right) \boldsymbol{v}_2$$

i.e. the direction that maximizes the variance among all other dimensions

This is just the second top eigenvector of the covariance matrix!

Finding the other PCs

If v_1 is the first PC, then the **second PC** is found via

$$\max_{\boldsymbol{v}_2 : \|\boldsymbol{v}_2\|_2 = 1, \boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_2 = 0} \boldsymbol{v}_2^{\mathrm{T}} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \right) \boldsymbol{v}_2$$

i.e. the direction that maximizes the variance among all other dimensions

This is just the second top eigenvector of the covariance matrix!

Conclusion: the d-th principal component is the d-th eigenvector (sorted by the eigenvalue from largest to smallest).

PCA

Input: a dataset represented as $\boldsymbol{X}$, $\#$components p we want

PCA

Input: a dataset represented as $\boldsymbol{X}$, $\#$components p we want

Step 1 Center the data by subtracting the mean

PCA

Input: a dataset represented as $\boldsymbol{X}$, $\#$components p we want

Step 1 Center the data by subtracting the mean

Step 2 Find the top p eigenvectors (with unit norm) of the covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$, denoted by $\boldsymbol{V} \in \mathbb{R}^{\mathsf{D}\times p}$

PCA

Input: a dataset represented as $\boldsymbol{X}$, $\#$components p we want

Step 1 Center the data by subtracting the mean

Step 2 Find the top p eigenvectors (with unit norm) of the covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$, denoted by $\boldsymbol{V} \in \mathbb{R}^{\mathsf{D}\times p}$

Step 3 Construct the new compressed dataset $\boldsymbol{X}\boldsymbol{V} \in \mathbb{R}^{N\times p}$

PCA

Input: a dataset represented as $\boldsymbol{X}$, $\#$components p we want

Step 1 Center the data by subtracting the mean

Step 2 Find the top p eigenvectors (with unit norm) of the covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$, denoted by $\boldsymbol{V} \in \mathbb{R}^{\mathsf{D}\times p}$

Step 3 Construct the new compressed dataset $\boldsymbol{X}\boldsymbol{V} \in \mathbb{R}^{N\times p}$

(Optional) Step 4 "Reconstruct" original dataset as $\boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^{\mathrm{T}} \in \mathbb{R}^{N\times \mathsf{D}}$

How many PCs do we want?

One common rule: pick p large enough so it **covers about 90% of the spectrum** (so 90% of variance is explained),

How many PCs do we want?

One common rule: pick p large enough so it **covers about 90% of the spectrum** (so 90% of variance is explained), i.e.

$$\frac{\sum_{d=1}^{p} \lambda_d}{\sum_{d=1}^{\mathsf{D}} \lambda_d} \geq 90\%$$

where $\lambda_1 \geq \cdots \geq \lambda_N$ are sorted eigenvalues.

How many PCs do we want?

One common rule: pick p large enough so it **covers about 90% of the spectrum** (so 90% of variance is explained), i.e.

$$\frac{\sum_{d=1}^{p} \lambda_d}{\sum_{d=1}^{\mathsf{D}} \lambda_d} \geq 90\%$$

where $\lambda_1 \geq \cdots \geq \lambda_N$ are sorted eigenvalues.

Note: $\sum_{d=1}^{\mathsf{D}} \lambda_d = \mathsf{Tr}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})$, so no need to actually find all eigenvalues.

How many PCs do we want?

One common rule: pick p large enough so it **covers about 90% of the spectrum** (so 90% of variance is explained), i.e.

$$\frac{\sum_{d=1}^{p} \lambda_d}{\sum_{d=1}^{\mathsf{D}} \lambda_d} \geq 90\%$$

where $\lambda_1 \geq \cdots \geq \lambda_N$ are sorted eigenvalues.

Note: $\sum_{d=1}^{\mathsf{D}} \lambda_d = \mathrm{Tr}(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X})$, so no need to actually find all eigenvalues.

For visualization, also often pick $p = 1$ or $p = 2$.

Another visualization example

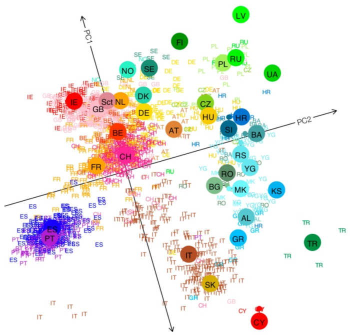
A famous study of **genetic map**

- dataset: genomes of 1,387 Europeans

Another visualization example

A famous study of **genetic map**

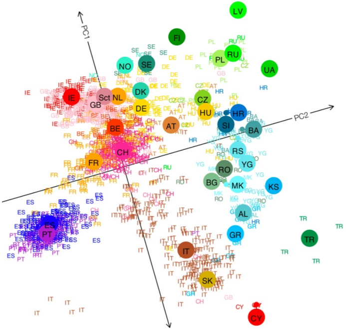
- dataset: genomes of 1,387 Europeans
- first 2 PCs shown below;

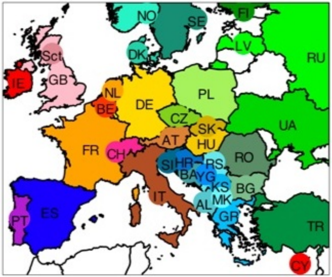


Another visualization example

A famous study of **genetic map**

- dataset: genomes of 1,387 Europeans
- first 2 PCs shown below; *looks remarkably like the geographic map*

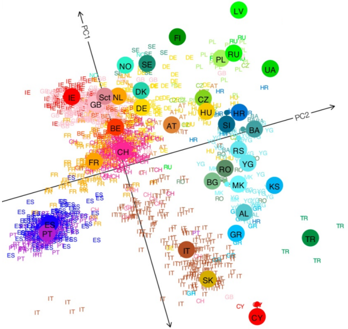


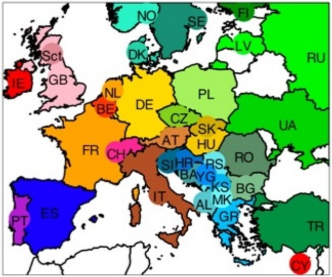


Another visualization example

A famous study of **genetic map**

- dataset: genomes of 1,387 Europeans
- first 2 PCs shown below; *looks remarkably like the geographic map*





Genetic variation is highly structured by geography; top PCs correspond to geographic factors.

Compression Example

Dataset: 256×256 ($\approx 65K$ pixels) dimensional images of about 2500 faces, all framed similarly ($N = 2500$, D $\approx 65,000$)

Compression Example

Dataset: 256×256 ($\approx 65K$ pixels) dimensional images of about 2500 faces, all framed similarly ($N = 2500$, D $\approx 65,000$)

Even with $p \approx 100$, reconstructed images $\boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^{\mathrm{T}} \in \mathbb{R}^{N \times \mathsf{D}}$ look very much similar to original images

Compression Example

Dataset: 256×256 ($\approx 65K$ pixels) dimensional images of about 2500 faces, all framed similarly ($N = 2500$, D $\approx 65,000$)

Even with $p \approx 100$, reconstructed images $\boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^{\mathrm{T}} \in \mathbb{R}^{N \times \mathsf{D}}$ look very much similar to original images

The principal components (called **eigenfaces** $\in \mathbb{R}^{\mathsf{D}}$) are themselves interpretable too!

Figure 2. Seven of the eigenfaces calculated from the input images of Figure 1.

Compression Example

Dataset: 256×256 ($\approx 65K$ pixels) dimensional images of about 2500 faces, all framed similarly ($N = 2500$, D $\approx 65,000$)

Even with $p \approx 100$, reconstructed images $\boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^{\mathrm{T}} \in \mathbb{R}^{N \times \mathsf{D}}$ look very much similar to original images

The principal components (called **eigenfaces** $\in \mathbb{R}^{\mathsf{D}}$) are themselves interpretable too!

PCA finds the subspace (out of the space of all 256×256 images) where faces lie

Figure 2. Seven of the eigenfaces calculated from the input images of Figure 1.

Word embedding via PCA

Create **meaningful vector representation** of words; see project.

Outline

Sequential data and Markov models

Sequential data (i.e., each $\boldsymbol{x}_n$ is a *sequence*) are ubiquitous:

- text or speech data
- stock market data
- gene data

Sequential data and Markov models

Sequential data (i.e., each $\boldsymbol{x}_n$ is a *sequence*) are ubiquitous:

- text or speech data
- stock market data
- gene data

Markov models are powerful probabilistic tools to analyze them:

Sequential data and Markov models

Sequential data (i.e., each $\boldsymbol{x}_n$ is a *sequence*) are ubiquitous:

- text or speech data
- stock market data
- gene data

Markov models are powerful probabilistic tools to analyze them:

- not the state-of-the-art for e.g. Natural Language Processing (NLP)

Sequential data and Markov models

Sequential data (i.e., each $\boldsymbol{x}_n$ is a *sequence*) are ubiquitous:

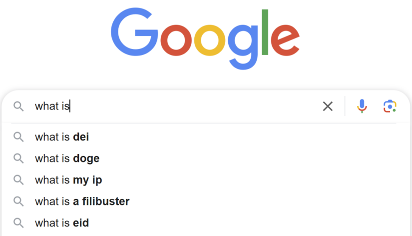
- text or speech data
- stock market data
- gene data

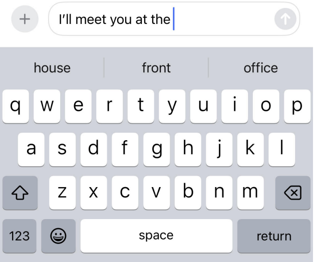
Markov models are powerful probabilistic tools to analyze them:

- not the state-of-the-art for e.g. Natural Language Processing (NLP)
- but still extremely useful in many areas (in particular, it is the foundation of *reinforcement learning*)

A motivating example

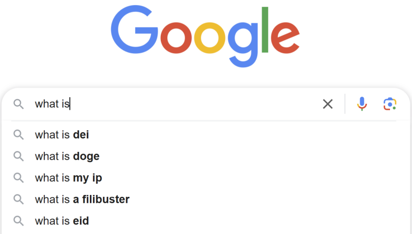
Next word prediction

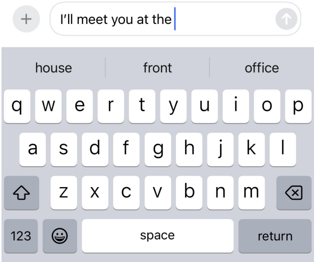




A motivating example

Next word prediction



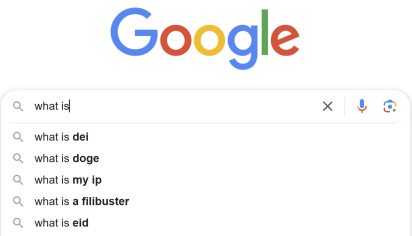


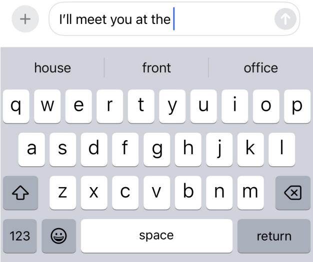
Can be formulated as predicting the probability of the next word/state:

$$P(Z_{t+1} \mid Z_1, \ldots, Z_t)?$$

A motivating example

Next word prediction





Can be formulated as predicting the probability of the next word/state:

$$P(Z_{t+1} \mid Z_1, \ldots, Z_t)?$$

abbreviated as $P(Z_{t+1} \mid Z_{1:t})$ (i.e., $Z_{1:t}$ denotes the sequence $Z_1, \ldots, Z_t$).

Definition of a Markov model/chain

A **Markov chain** is a stochastic process with **Markov property**:

Definition of a Markov model/chain

A **Markov chain** is a stochastic process with **Markov property**: a sequence of random variables $Z_1, Z_2, \cdots$ s.t.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t) \qquad \text{(Markov property)}$$

i.e. *the current state only depends on the most recent state*

Definition of a Markov model/chain

A **Markov chain** is a stochastic process with **Markov property**: a sequence of random variables $Z_1, Z_2, \cdots$ s.t.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t) \qquad \text{(Markov property)}$$

i.e. *the current state only depends on the most recent state*

We only consider the following case:

- All Z_t's take value from the same discrete set $\{1, \ldots, S\}$

Definition of a Markov model/chain

A **Markov chain** is a stochastic process with **Markov property**: a sequence of random variables $Z_1, Z_2, \cdots$ s.t.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t) \qquad \text{(Markov property)}$$

i.e. *the current state only depends on the most recent state*

We only consider the following case:

- All Z_t's take value from the same discrete set $\{1, \ldots, S\}$
- $P(Z_{t+1} = s' \mid Z_t = s) = a_{s,s'}$, known as transition probability

Definition of a Markov model/chain

A **Markov chain** is a stochastic process with **Markov property**: a sequence of random variables $Z_1, Z_2, \cdots$ s.t.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t) \qquad \text{(Markov property)}$$

i.e. *the current state only depends on the most recent state*

We only consider the following case:

- All Z_t's take value from the same discrete set $\{1, \ldots, S\}$
- $P(Z_{t+1} = s' \mid Z_t = s) = a_{s,s'}$, known as transition probability
- $P(Z_1 = s) = \pi_s$

Definition of a Markov model/chain

A **Markov chain** is a stochastic process with **Markov property**: a sequence of random variables $Z_1, Z_2, \cdots$ s.t.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t) \qquad \text{(Markov property)}$$

i.e. *the current state only depends on the most recent state*

We only consider the following case:

- All Z_t's take value from the same discrete set $\{1, \ldots, S\}$
- $P(Z_{t+1} = s' \mid Z_t = s) = a_{s,s'}$, known as transition probability
- $P(Z_1 = s) = \pi_s$
- $(\{\pi_s\}, \{a_{s,s'}\}) = (\boldsymbol{\pi}, \boldsymbol{A})$ are parameters of the model

Examples

- Example 1 (**Language model**)

  States $[S]$ represent a dictionary of words,

$$a_{\text{ice,cream}} = P(Z_{t+1} = \text{cream} \mid Z_t = \text{ice})$$

is an example of the transition probability.

Examples

- Example 1 (**Language model**)

  States $[S]$ represent a dictionary of words,

$$a_{\text{ice,cream}} = P(Z_{t+1} = \text{cream} \mid Z_t = \text{ice})$$

  is an example of the transition probability.

- Example 2 (**Weather**)

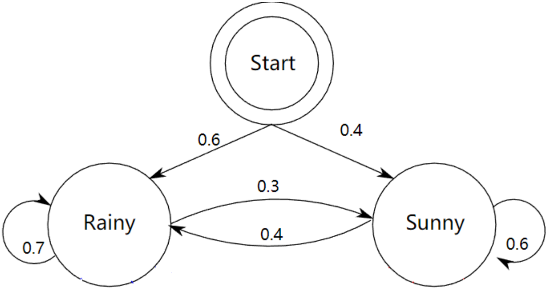
  States $[S]$ represent weather at each day

$$a_{\text{sunny,rainy}} = P(Z_{t+1} = \text{rainy} \mid Z_t = \text{sunny})$$

Graph representation

picture from Wikipedia

It is intuitive to represent a Markov model as a **graph**



High-order Markov chain

Is the Markov assumption reasonable?

High-order Markov chain

Is the Markov assumption reasonable? Not completely for the language model for example:

$$P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the})$$
$$= P(Z_{t+1} = \text{office} \mid Z_t = \text{the}) \qquad (\textbf{bigram}, \text{ unreasonable})$$

High-order Markov chain

Is the Markov assumption reasonable? Not completely for the language model for example:

$$P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\ = P(Z_{t+1} = \text{office} \mid Z_t = \text{the}) \qquad (\textbf{bigram}, \text{ unreasonable})$$

Higher-order Markov chains make it more reasonable, e.g.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t, Z_{t-1}) \qquad (\textbf{second-order Markov})$$

High-order Markov chain

Is the Markov assumption reasonable? Not completely for the language model for example:

$$P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the})$$
$$= P(Z_{t+1} = \text{office} \mid Z_t = \text{the}) \qquad \text{(\textbf{bigram}, unreasonable)}$$

Higher-order Markov chains make it more reasonable, e.g.

$$P(Z_{t+1} \mid Z_{1:t}) = P(Z_{t+1} \mid Z_t, Z_{t-1}) \qquad \text{(\textbf{second-order Markov})}$$

i.e. the current word only depends on the last two words (the **trigram** model).

n-gram model

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-1:t} = \text{at the}) \qquad (\textbf{trigram})
\end{aligned}
$$

n-gram model

$$
\begin{aligned}
&P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
&= P(Z_{t+1} = \text{office} \mid Z_{t-1:t} = \text{at the}) && \textbf{(trigram)}
\end{aligned}
$$

$$
\begin{aligned}
&P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
&= P(Z_{t+1} = \text{office} \mid Z_{t-2:t} = \text{you at the}) && \textbf{(4-gram)}
\end{aligned}
$$

n-gram model

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-1:t} = \text{at the}) && \textbf{(trigram)}
\end{aligned}
$$

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-2:t} = \text{you at the}) && \textbf{(4-gram)}
\end{aligned}
$$

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-3:t} = \text{meet you at the}) && \textbf{(5-gram)}
\end{aligned}
$$

n-gram model

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-1:t} = \text{at the}) && (\textbf{trigram})
\end{aligned}
$$

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-2:t} = \text{you at the}) && (\textbf{4-gram})
\end{aligned}
$$

$$
\begin{aligned}
& P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
= {} & P(Z_{t+1} = \text{office} \mid Z_{t-3:t} = \text{meet you at the}) && (\textbf{5-gram})
\end{aligned}
$$

Learning higher-order Markov chains is similar, but more *expensive*.

n-gram model

$$
\begin{aligned}
&P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
&= P(Z_{t+1} = \text{office} \mid Z_{t-1:t} = \text{at the}) \qquad \textbf{(trigram)}
\end{aligned}
$$

$$
\begin{aligned}
&P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
&= P(Z_{t+1} = \text{office} \mid Z_{t-2:t} = \text{you at the}) \qquad \textbf{(4-gram)}
\end{aligned}
$$

$$
\begin{aligned}
&P(Z_{t+1} = \text{office} \mid Z_{1:t} = \text{I'll meet you at the}) \\
&= P(Z_{t+1} = \text{office} \mid Z_{t-3:t} = \text{meet you at the}) \qquad \textbf{(5-gram)}
\end{aligned}
$$

Learning higher-order Markov chains is similar, but more *expensive*.

For simplicity, we will only consider standard Markov chains.

Learning from examples

Now suppose we have observed N **sequences of examples**:

- $z_{1,1}, \ldots, z_{1,T}$
- $\cdots$
- $z_{n,1}, \ldots, z_{n,T}$
- $\cdots$
- $z_{N,1}, \ldots, z_{N,T}$

Learning from examples

Now suppose we have observed N **sequences of examples**:

- $z_{1,1}, \ldots, z_{1,T}$
- $\cdots$
- $z_{n,1}, \ldots, z_{n,T}$
- $\cdots$
- $z_{N,1}, \ldots, z_{N,T}$

where

- for simplicity we assume each sequence has the same length T

Learning from examples

Now suppose we have observed N **sequences of examples**:

- $z_{1,1}, \ldots, z_{1,T}$
- $\cdots$
- $z_{n,1}, \ldots, z_{n,T}$
- $\cdots$
- $z_{N,1}, \ldots, z_{N,T}$

where

- for simplicity we assume each sequence has the same length T
- lower case $z_{n,t}$ represents the value of the random variable $Z_{n,t}$

Learning from examples

Now suppose we have observed N **sequences of examples**:

- $z_{1,1}, \ldots, z_{1,T}$
- $\cdots$
- $z_{n,1}, \ldots, z_{n,T}$
- $\cdots$
- $z_{N,1}, \ldots, z_{N,T}$

where

- for simplicity we assume each sequence has the same length T
- lower case $z_{n,t}$ represents the value of the random variable $Z_{n,t}$

From these observations how do we *learn the model parameters* $(\boldsymbol{\pi}, \boldsymbol{A})$?

Finding the MLE

Same story, find the **MLE**.

Finding the MLE

Same story, find the **MLE**. The log-likelihood of a sequence $z_1, \ldots, z_T$ is

$$\ln P(Z_{1:T} = z_{1:T})$$

Finding the MLE

Same story, find the **MLE**. The log-likelihood of a sequence $z_1, \ldots, z_T$ is

$$\begin{aligned} &\ln P(Z_{1:T} = z_{1:T}) \\ &= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{1:t-1} = z_{1:t-1}) && \text{(always true)} \end{aligned}$$

Finding the MLE

Same story, find the **MLE**. The log-likelihood of a sequence $z_1, \ldots, z_T$ is

$$
\begin{aligned}
&\ln P(Z_{1:T} = z_{1:T}) \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{1:t-1} = z_{1:t-1}) && \text{(always true)} \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{t-1} = z_{t-1}) && \text{(Markov property)}
\end{aligned}
$$

Finding the MLE

Same story, find the **MLE**. The log-likelihood of a sequence $z_1, \ldots, z_T$ is

$$
\begin{aligned}
&\ln P(Z_{1:T} = z_{1:T}) \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{1:t-1} = z_{1:t-1}) && \text{(always true)} \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{t-1} = z_{t-1}) && \text{(Markov property)} \\
&= \ln \pi_{z_1} + \sum_{t=2}^{T} \ln a_{z_{t-1}, z_t}
\end{aligned}
$$

Finding the MLE

Same story, find the **MLE**. The log-likelihood of a sequence $z_1, \ldots, z_T$ is

$$
\begin{aligned}
& \ln P(Z_{1:T} = z_{1:T}) \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{1:t-1} = z_{1:t-1}) && \text{(always true)} \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{t-1} = z_{t-1}) && \text{(Markov property)} \\
&= \ln \pi_{z_1} + \sum_{t=2}^{T} \ln a_{z_{t-1}, z_t} \\
&= \sum_{s} \mathbb{I}[z_1 = s] \ln \pi_s + \sum_{s,s'} \left(\sum_{t=2}^{T} \mathbb{I}[z_{t-1} = s, z_t = s'] \right) \ln a_{s,s'}
\end{aligned}
$$

Finding the MLE

Summing over all N sequences, we obtain the joint log-likelihood:

$$L(\boldsymbol{\pi}, \boldsymbol{A}) = \sum_{s} (\textbf{\#initial states with value } s) \ln \pi_s + \sum_{s,s'} (\textbf{\#transitions from } s \textbf{ to } s') \ln a_{s,s'}$$

Finding the MLE

Summing over all N sequences, we obtain the joint log-likelihood:

$$L(\boldsymbol{\pi}, \boldsymbol{A}) = \sum_s (\textbf{\#initial states with value } s) \ln \pi_s + \sum_{s,s'} (\textbf{\#transitions from } s \textbf{ to } s') \ln a_{s,s'}$$

The MLE $\operatorname{argmax}_{\boldsymbol{\pi},\boldsymbol{A}} L(\boldsymbol{\pi}, \boldsymbol{A})$ is:

$$\pi_s \propto \#\text{initial states with value } s$$
$$a_{s,s'} \propto \#\text{transitions from } s \text{ to } s'$$

Example

Suppose we observed the following 2 sequences of length 5

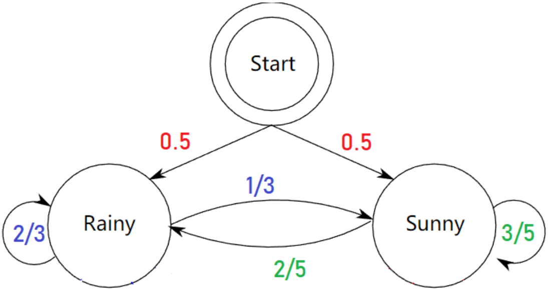
- sunny, sunny, rainy, rainy, rainy
- rainy, sunny, sunny, sunny, rainy

Example

Suppose we observed the following 2 sequences of length 5

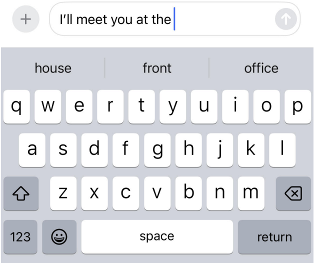
- sunny, sunny, rainy, rainy, rainy
- rainy, sunny, sunny, sunny, rainy

MLE is the following model



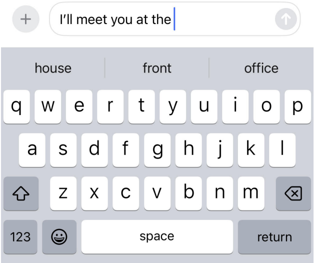
n-gram example

Recall: I'll meet you at the ______.



n-gram example

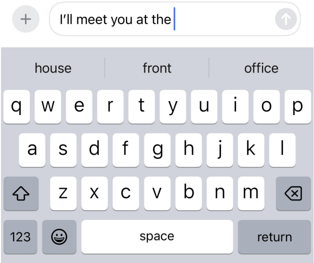
Recall: I'll meet you at the ______.



Suppose

- we use a 5-gram model

n-gram example

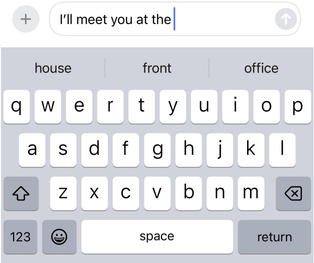


Recall: I'll meet you at the _____.

Suppose

- we use a 5-gram model
- "meet you at the" appears 1000 times in the training set:
  - "meet you at the house" appears 500 times
  - "meet you at the front" appears 300 times
  - "meet you at the office" appears 200 times

n-gram example



Recall: I'll meet you at the ______.

Suppose

- we use a 5-gram model
- "meet you at the" appears 1000 times in the training set:
  - "meet you at the house" appears 500 times
  - "meet you at the front" appears 300 times
  - "meet you at the office" appears 200 times

Then the model predicts the next word as "house" with prob. 0.5, "front" with prob. 0.3, and "office" with prob. 0.2.

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

- today the ______

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

- today the ______
- today the price ______

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

- today the ______
- today the price ______
- today the price of ______

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

of	0.308
for	0.050
it	0.046
to	0.046
is	0.031
…	

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

- today the ______
- today the price ______
- today the price of ______
- today the price of gold ______

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

of	0.308
for	0.050
it	0.046
to	0.046
is	0.031
…	

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
…	

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

- today the ______
- today the price ______
- today the price of ______
- today the price of gold ______

company	0.153
bank	0.153
$\boxed{\text{price}}$	0.077
italian	0.039
emirate	0.039
…	

$\boxed{\text{of}}$	0.308
for	0.050
it	0.046
to	0.046
is	0.031
…	

the	0.072
18	0.043
oil	0.043
its	0.036
$\boxed{\text{gold}}$	0.018
…	

final result: *today the price of gold per ton, while production of shoe lasts and shoe industry, the bank intervened just after it considered and rejected an IMF demand to rebuild depleted European stocks.*

Text generation

adapted from Stanford CS224n

After learning the model, can also **generate texts** by repeatedly sampling conditioning on what have been generated:

- today the ______
- today the price ______
- today the price of ______
- today the price of gold ______

company	0.153
bank	0.153
$\boxed{\text{price}}$	$\boxed{0.077}$
italian	0.039
emirate	0.039
…	

$\boxed{\text{of}}$	$\boxed{0.308}$
for	0.050
it	0.046
to	0.046
is	0.031
…	

the	0.072
18	0.043
oil	0.043
its	0.036
$\boxed{\text{gold}}$	$\boxed{0.018}$
…	

final result: *today the price of gold per ton, while production of shoe lasts and shoe industry, the bank intervened just after it considered and rejected an IMF demand to rebuild depleted European stocks.*

Surprisingly grammatical! but *incoherent*...

Outline

Markov model with outcomes/observations

Now suppose each state Z_t also "emits" some **outcome/observation** $X_t \in [O]$ based on the following model

$$P(X_t = o \mid Z_t = s) = b_{s,o} \qquad \text{(emission probability)}$$

independent of anything else.

Markov model with outcomes/observations

Now suppose each state Z_t also "emits" some **outcome/observation** $X_t \in [O]$ based on the following model

$$P(X_t = o \mid Z_t = s) = b_{s,o} \qquad \text{(emission probability)}$$

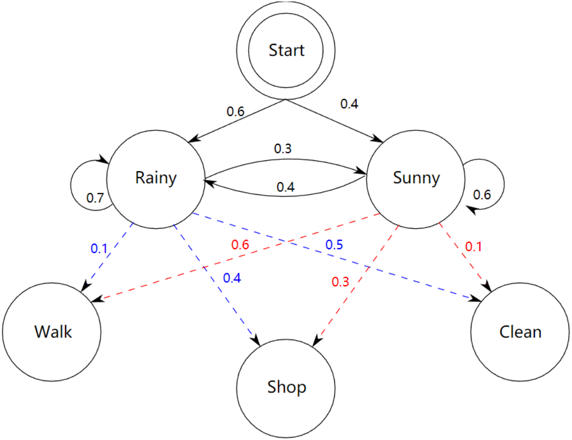
independent of anything else.

The model parameters are $(\{\pi_s\}, \{a_{s,s'}\}, \{b_{s,o}\}) = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.

Another (toy) running example

picture from Wikipedia

On each day, we also observe **Bob's activity: walk, shop, or clean**, which only depends on the weather of that day.



Joint likelihood

The joint log-likelihood of a state-outcome sequence $z_1, x_1, \ldots, z_T, x_T$ is

$$\ln P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T})$$

Joint likelihood

The joint log-likelihood of a state-outcome sequence $z_1, x_1, \ldots, z_T, x_T$ is

$$\begin{aligned}
&\ln P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) \\
&= \ln P(Z_{1:T} = z_{1:T}) + \ln P(X_{1:T} = x_{1:T} \mid Z_{1:T} = z_{1:T}) \quad \text{(always true)}
\end{aligned}$$

Joint likelihood

The joint log-likelihood of a state-outcome sequence $z_1, x_1, \ldots, z_T, x_T$ is

$$
\begin{aligned}
&\ln P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) \\
&= \ln P(Z_{1:T} = z_{1:T}) + \ln P(X_{1:T} = x_{1:T} \mid Z_{1:T} = z_{1:T}) \quad \text{(always true)} \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{t-1} = z_{t-1}) + \sum_{t=1}^{T} \ln P(X_t = x_t \mid Z_t = z_t)
\end{aligned}
$$

(due to all the independence)

Joint likelihood

The joint log-likelihood of a state-outcome sequence $z_1, x_1, \ldots, z_T, x_T$ is

$$\begin{aligned}
&\ln P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) \\
&= \ln P(Z_{1:T} = z_{1:T}) + \ln P(X_{1:T} = x_{1:T} \mid Z_{1:T} = z_{1:T}) \quad \text{(always true)} \\
&= \sum_{t=1}^{T} \ln P(Z_t = z_t \mid Z_{t-1} = z_{t-1}) + \sum_{t=1}^{T} \ln P(X_t = x_t \mid Z_t = z_t) \\
&\qquad \text{(due to all the independence)} \\
&= \ln \pi_{z_1} + \sum_{t=2}^{T} \ln a_{z_{t-1}, z_t} + \sum_{t=1}^{T} \ln b_{z_t, x_t}
\end{aligned}$$

Learning the model

If we observe N state-outcome sequences: $z_{n,1}, x_{n,1}, \ldots, z_{n,T}, x_{n,T}$ for $n = 1, \ldots, N$, the MLE is again very simple (verify yourself):

$$\pi_s \propto \textbf{\#initial states with value } s$$
$$a_{s,s'} \propto \textbf{\#transitions from } s \textbf{ to } s'$$
$$b_{s,o} \propto \textbf{\#state-outcome pairs } (s, o)$$

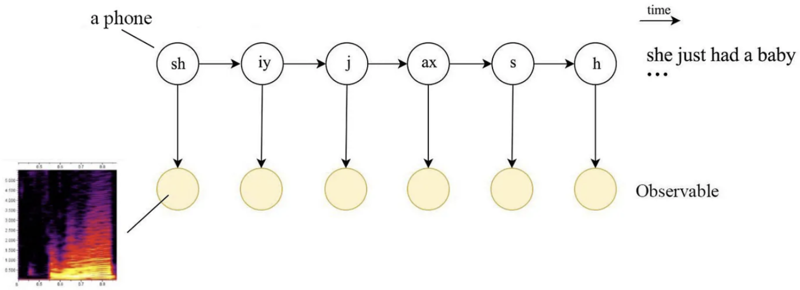
Hidden Markov models

However, *most often we do not observe the states!* Hence the name **Hidden Markov Model (HMM)**

Hidden Markov models

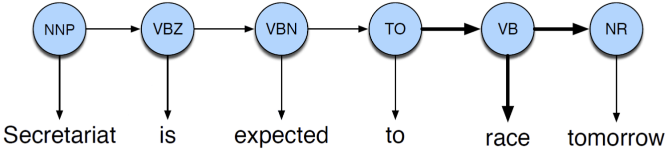
However, *most often we do not observe the states!* Hence the name **Hidden Markov Model (HMM)**

- **speech recognition**: observe the speech $X_{1:T}$ but not the underlying words/phones $Z_{1:T}$



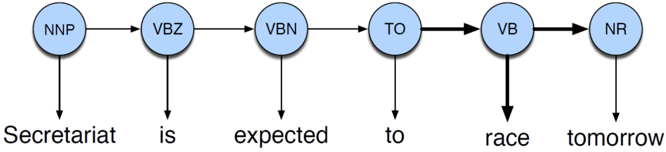
Hidden Markov models (cont.)

- **Part of Speech (POS) tagging**: observe the sentence $X_{1:T}$ but not the underlying categories $Z_{1:T}$ (noun, verb, adjective, adverb, etc.)



Hidden Markov models (cont.)

- **Part of Speech (POS) tagging**: observe the sentence $X_{1:T}$ but not the underlying categories $Z_{1:T}$ (noun, verb, adjective, adverb, etc.)



See programming project!

Learning HMMs

How to learn HMMs?

Learning HMMs

How to learn HMMs? **Roadmap**:

- first discuss how to **infer** when the model is known (key: dynamic programming)

Learning HMMs

How to learn HMMs? **Roadmap**:

- first discuss how to **infer** when the model is known (key: dynamic programming)
- then discuss how to **learn** the model (key: EM)

What can we infer about an HMM?

Knowing the parameter of an HMM, we can infer

- **most likely hidden states path, given an observation sequence**

$$\operatorname*{argmax}_{z_{1:T}} P(Z_{1:T} = z_{1:T} \mid X_{1:T} = x_{1:T})$$

  e.g. given Bob's activities for one week, what's the most likely weather for this week?

What can we infer about an HMM?

Knowing the parameter of an HMM, we can infer

- **most likely hidden states path, given an observation sequence**

$$\underset{z_{1:T}}{\operatorname{argmax}}\, P(Z_{1:T} = z_{1:T} \mid X_{1:T} = x_{1:T})$$

  e.g. given Bob's activities for one week, what's the most likely weather for this week?

- **the probability of observing some sequence**

$$P(X_{1:T} = x_{1:T})$$

  e.g. prob. of observing Bob's activities "walk, walk, shop, clean, walk, shop, shop" for one week

What can we infer for a known HMM?

Knowing the parameter of an HMM, we can infer

- **the state at some point, given an observation sequence**

$$P(Z_t = s \mid X_{1:T} = x_{1:T})$$

  e.g. given Bob's activities for one week, how was the weather like on Wed?

What can we infer for a known HMM?

Knowing the parameter of an HMM, we can infer

- **the state at some point, given an observation sequence**

$$P(Z_t = s \mid X_{1:T} = x_{1:T})$$

  e.g. given Bob's activities for one week, how was the weather like on Wed?

- **the transition at some point, given an observation sequence**

$$P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T})$$

  e.g. given Bob's activities for one week, how was the weather like on Wed and Thu?

Forward and backward messages

The key to infer all these is to compute two things:

Forward and backward messages

The key to infer all these is to compute two things:

- **forward messages**: for each s and t

$$\alpha_s(t) = P(Z_t = s, X_{1:t} = x_{1:t})$$

Forward and backward messages

The key to infer all these is to compute two things:

- **forward messages**: for each s and t

$$\alpha_s(t) = P(Z_t = s, X_{1:t} = x_{1:t})$$

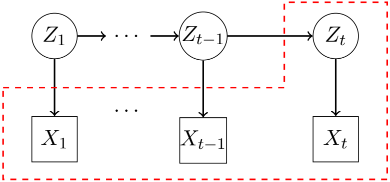
- **backward messages**: for each s and t

$$\beta_s(t) = P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s)$$

Computing forward messages

Key: *establish recursion*

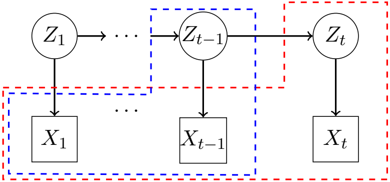
$$\alpha_s(t) = P(Z_t = s, X_{1:t} = x_{1:t})$$



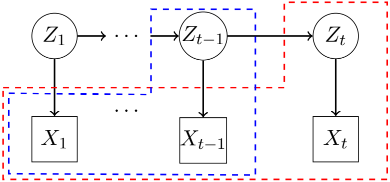
Computing forward messages

Key: *establish recursion*

$$\alpha_s(t) = P(Z_t = s, X_{1:t} = x_{1:t})$$



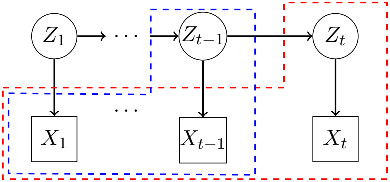
Computing forward messages



Key: *establish recursion*

$$
\begin{aligned}
&\alpha_s(t) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) \\
&= \sum_{s'} P(Z_t = s, Z_{t-1} = s', X_{1:t} = x_{1:t}) \qquad \text{(marginalizing)}
\end{aligned}
$$

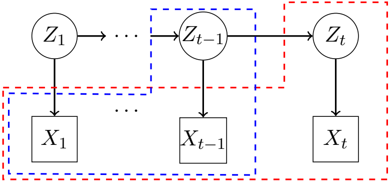
Computing forward messages



Key: *establish recursion*

$$
\begin{aligned}
&\alpha_s(t) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) \\
&= \sum_{s'} P(Z_t = s, Z_{t-1} = s', X_{1:t} = x_{1:t}) \qquad \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t-1} = s', X_{1:t-1} = x_{1:t-1}) \\
&\qquad \times P(Z_t = s, X_t = x_t \mid Z_{t-1} = s', X_{1:t-1} = x_{1:t-1})
\end{aligned}
$$

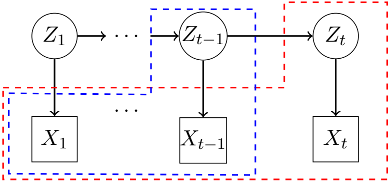
Computing forward messages



Key: *establish recursion*

$$
\begin{aligned}
&\alpha_s(t) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) \\
&= \sum_{s'} P(Z_t = s, Z_{t-1} = s', X_{1:t} = x_{1:t}) && \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t-1} = s', X_{1:t-1} = x_{1:t-1}) \\
&\qquad\qquad \times P(Z_t = s, X_t = x_t \mid Z_{t-1} = s', X_{1:t-1} = x_{1:t-1}) \\
&= b_{s,x_t} \sum_{s'} a_{s',s} \alpha_{s'}(t-1) && \text{(by the model definition)}
\end{aligned}
$$

Computing forward messages



Key: *establish recursion*

$$
\begin{aligned}
&\alpha_s(t) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) \\
&= \sum_{s'} P(Z_t = s, Z_{t-1} = s', X_{1:t} = x_{1:t}) && \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t-1} = s', X_{1:t-1} = x_{1:t-1}) \\
&\qquad \times P(Z_t = s, X_t = x_t \mid Z_{t-1} = s', X_{1:t-1} = x_{1:t-1}) \\
&= b_{s,x_t} \sum_{s'} a_{s',s} \alpha_{s'}(t-1) && \text{(by the model definition)}
\end{aligned}
$$

Base case: $\alpha_s(1) = P(Z_1 = s, X_1 = x_1) = \pi_s b_{s,x_1}$

Forward procedure

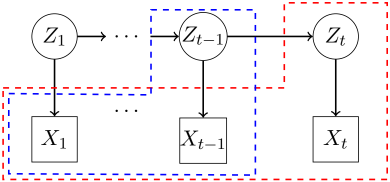
Forward procedure

For all $s \in [S]$, compute $\alpha_s(1) = \pi_s b_{s,x_1}$.

For $t = 2, \ldots, T$

- for each $s \in [S]$, compute

$$\alpha_s(t) = b_{s,x_t} \sum_{s'} a_{s',s} \alpha_{s'}(t-1)$$



Forward procedure

Forward procedure

For all $s \in [S]$, compute $\alpha_s(1) = \pi_s b_{s,x_1}$.

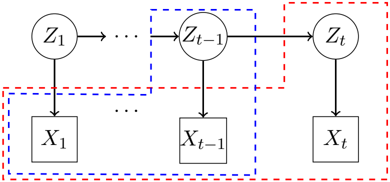
For $t = 2, \ldots, T$

- for each $s \in [S]$, compute

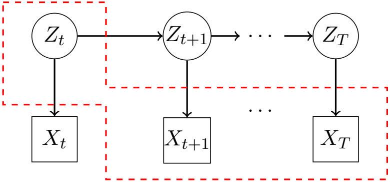
$$\alpha_s(t) = b_{s,x_t} \sum_{s'} a_{s',s} \alpha_{s'}(t-1)$$

It takes

- $O(S^2T)$ time
- $O(ST)$ space



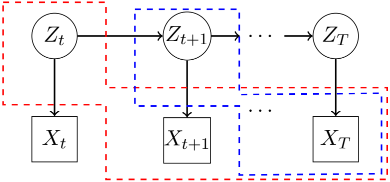
Computing backward messages



Again, establish a recursion

$$\beta_s(t)$$
$$= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s)$$

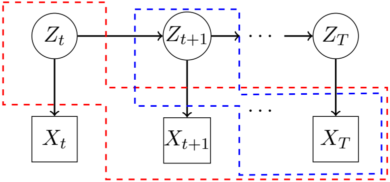
Computing backward messages



Again, establish a recursion

$$\beta_s(t)$$
$$= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s)$$

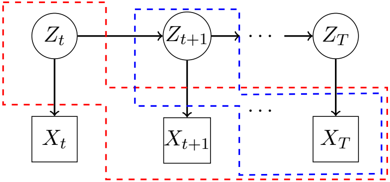
Computing backward messages



Again, establish a recursion

$$\begin{aligned}
&\beta_s(t) \\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s) \\
&= \sum_{s'} P(X_{t+1:T} = x_{t+1:T}, Z_{t+1} = s' \mid Z_t = s) \qquad \text{(marginalizing)}
\end{aligned}$$

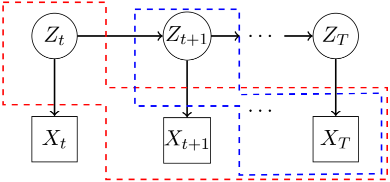
Computing backward messages



Again, establish a recursion

$$
\begin{aligned}
&\beta_s(t) \\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s) \\
&= \sum_{s'} P(X_{t+1:T} = x_{t+1:T}, Z_{t+1} = s' \mid Z_t = s) \qquad \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s', Z_t = s)
\end{aligned}
$$

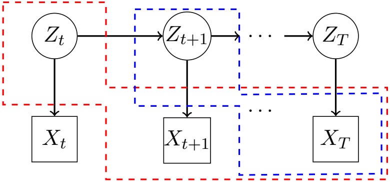
Computing backward messages



Again, establish a recursion

$$
\begin{aligned}
&\beta_s(t) \\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s) \\
&= \sum_{s'} P(X_{t+1:T} = x_{t+1:T}, Z_{t+1} = s' \mid Z_t = s) \qquad \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s', Z_t = s) \\
&= \sum_{s'} a_{s,s'} P(X_{t+1} = x_{t+1} \mid Z_{t+1} = s') P(X_{t+2:T} = x_{t+2:T} \mid Z_{t+1} = s')
\end{aligned}
$$

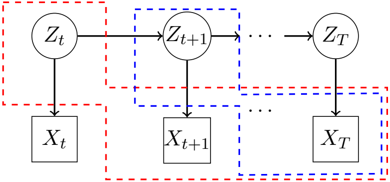
Computing backward messages



Again, establish a recursion

$$\begin{aligned}
&\beta_s(t) \\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s) \\
&= \sum_{s'} P(X_{t+1:T} = x_{t+1:T}, Z_{t+1} = s' \mid Z_t = s) \qquad \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s', Z_t = s) \\
&= \sum_{s'} a_{s,s'} P(X_{t+1} = x_{t+1} \mid Z_{t+1} = s') P(X_{t+2:T} = x_{t+2:T} \mid Z_{t+1} = s') \\
&= \sum_{s'} a_{s,s'} b_{s',x_{t+1}} \beta_{s'}(t+1)
\end{aligned}$$

Computing backward messages



Again, establish a recursion

$$
\begin{aligned}
&\beta_s(t) \\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s) \\
&= \sum_{s'} P(X_{t+1:T} = x_{t+1:T}, Z_{t+1} = s' \mid Z_t = s) \qquad \text{(marginalizing)} \\
&= \sum_{s'} P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s', Z_t = s) \\
&= \sum_{s'} a_{s,s'} P(X_{t+1} = x_{t+1} \mid Z_{t+1} = s') P(X_{t+2:T} = x_{t+2:T} \mid Z_{t+1} = s') \\
&= \sum_{s'} a_{s,s'} b_{s',x_{t+1}} \beta_{s'}(t+1)
\end{aligned}
$$

Base case: $\beta_s(T) = 1$

Backward procedure

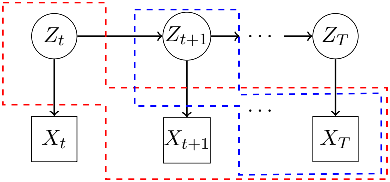
Backward procedure

For all $s \in [S]$, set $\beta_s(T) = 1$.

For $t = T-1, \ldots, 1$

- for each $s \in [S]$, compute

$$\beta_s(t) = \sum_{s'} a_{s,s'} b_{s',x_{t+1}} \beta_{s'}(t+1)$$



Backward procedure

Backward procedure

For all $s \in [S]$, set $\beta_s(T) = 1$.

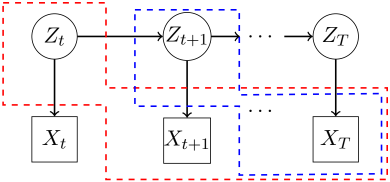
For $t = T-1, \ldots, 1$

- for each $s \in [S]$, compute

$$\beta_s(t) = \sum_{s'} a_{s,s'} b_{s',x_{t+1}} \beta_{s'}(t+1)$$

Again it takes

- $O(S^2T)$ time
- $O(ST)$ space

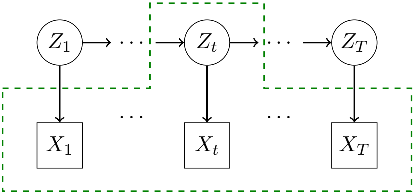


Using forward and backward messages

With forward and backward messages, we can easily infer many things,

Using forward and backward messages

With forward and backward messages, we can easily infer many things, e.g.

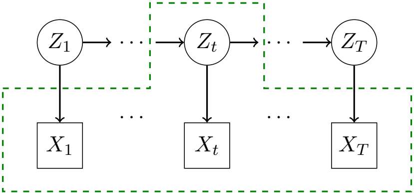


$$\gamma_s(t) = P(Z_t = s \mid X_{1:T} = x_{1:T})$$

Using forward and backward messages

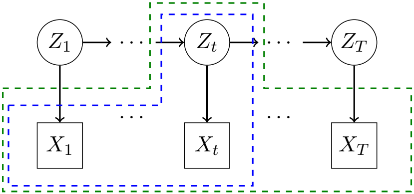
With forward and backward messages, we can easily infer many things, e.g.

$$\begin{aligned}\gamma_s(t) &= P(Z_t = s \mid X_{1:T} = x_{1:T}) \\ &\propto P(Z_t = s, X_{1:T} = x_{1:T})\end{aligned}$$



Using forward and backward messages

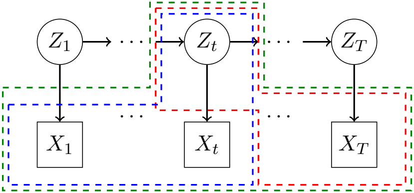
With forward and backward messages, we can easily infer many things, e.g.



$$\begin{aligned}
\gamma_s(t) &= P(Z_t = s \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t})
\end{aligned}$$

Using forward and backward messages

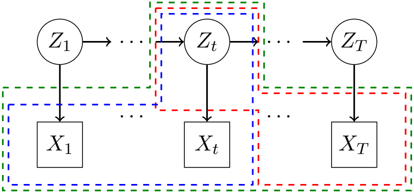
With forward and backward messages, we can easily infer many things, e.g.



$$
\begin{aligned}
\gamma_s(t) &= P(Z_t = s \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t}) \\
&= \alpha_s(t) \beta_s(t)
\end{aligned}
$$

Using forward and backward messages

With forward and backward messages, we can easily infer many things, e.g.

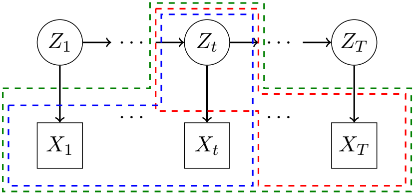


$$\begin{aligned}
\gamma_s(t) &= P(Z_t = s \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t}) \\
&= \alpha_s(t)\beta_s(t)
\end{aligned}$$

What constant are we omitting in "$\propto$"?

Using forward and backward messages

With forward and backward messages, we can easily infer many things, e.g.



$$
\begin{aligned}
\gamma_s(t) &= P(Z_t = s \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t}) \\
&= \alpha_s(t)\beta_s(t)
\end{aligned}
$$

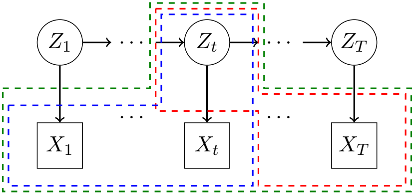
What constant are we omitting in "$\propto$"? It is exactly

$$P(X_{1:T} = x_{1:T}) = \sum_s \alpha_s(t)\beta_s(t),$$

the probability of observing the sequence $x_{1:T}$.

Using forward and backward messages

With forward and backward messages, we can easily infer many things, e.g.



$$
\begin{aligned}
\gamma_s(t) &= P(Z_t = s \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t}) \\
&= \alpha_s(t)\beta_s(t)
\end{aligned}
$$

What constant are we omitting in "$\propto$"? It is exactly

$$P(X_{1:T} = x_{1:T}) = \sum_s \alpha_s(t)\beta_s(t),$$

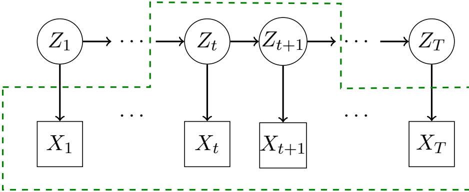
the probability of observing the sequence $x_{1:T}$.

This is true for any t; a good way to check correctness of your code.

Using forward and backward messages

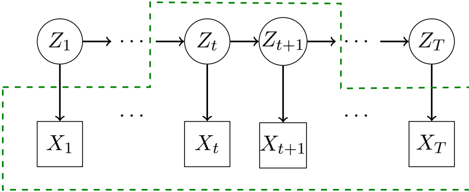
Another example: the conditional probability of transition s to s' at time t

$$\xi_{s,s'}(t) = P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T})$$



Using forward and backward messages

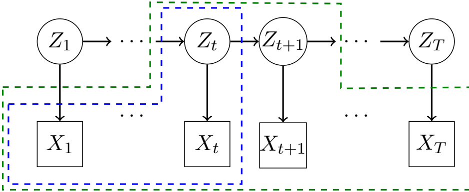
Another example: the conditional probability of transition s to s' at time t



$$\xi_{s,s'}(t)$$
$$= P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T})$$
$$\propto P(Z_t = s, Z_{t+1} = s', X_{1:T} = x_{1:T})$$

Using forward and backward messages

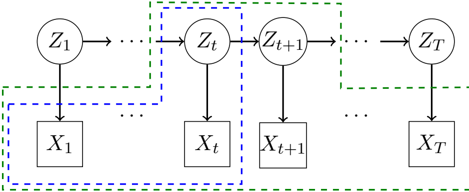
Another example: the conditional probability of transition s to s' at time t



$$\xi_{s,s'}(t)$$
$$= P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T})$$
$$\propto P(Z_t = s, Z_{t+1} = s', X_{1:T} = x_{1:T})$$
$$= P(Z_t = s, X_{1:t} = x_{1:t}) P(Z_{t+1} = s', X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t})$$

Using forward and backward messages

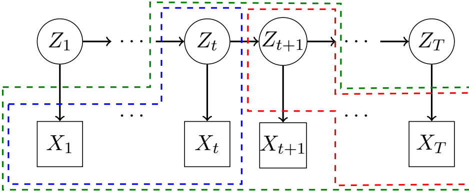
Another example: the conditional probability of transition s to s' at time t



$$
\begin{aligned}
&\xi_{s,s'}(t) \\
&= P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, Z_{t+1} = s', X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(Z_{t+1} = s', X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t}) \\
&= \alpha_s(t) P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s')
\end{aligned}
$$

Using forward and backward messages

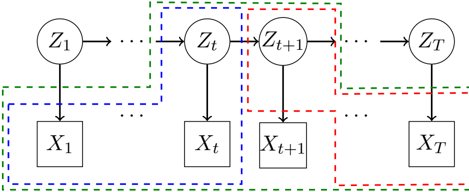
Another example: the conditional probability of transition s to s' at time t



$$
\begin{aligned}
&\xi_{s,s'}(t)\\
&= P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T})\\
&\propto P(Z_t = s, Z_{t+1} = s', X_{1:T} = x_{1:T})\\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(Z_{t+1} = s', X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t})\\
&= \alpha_s(t) P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s')\\
&= \alpha_s(t) a_{s,s'} P(X_{t+1} = x_{t+1} \mid Z_{t+1} = s') P(X_{t+2:T} = x_{t+2:T} \mid Z_{t+1} = s')
\end{aligned}
$$

Using forward and backward messages

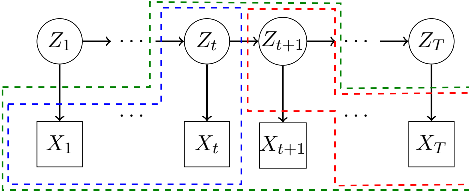
Another example: the conditional probability of transition s to s' at time t



$$\begin{aligned}
&\xi_{s,s'}(t) \\
&= P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T}) \\
&\propto P(Z_t = s, Z_{t+1} = s', X_{1:T} = x_{1:T}) \\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(Z_{t+1} = s', X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t}) \\
&= \alpha_s(t) P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s') \\
&= \alpha_s(t) a_{s,s'} P(X_{t+1} = x_{t+1} \mid Z_{t+1} = s') P(X_{t+2:T} = x_{t+2:T} \mid Z_{t+1} = s') \\
&= \alpha_s(t) a_{s,s'} b_{s',x_{t+1}} \beta_{s'}(t+1)
\end{aligned}$$

Using forward and backward messages

Another example: the conditional probability of transition s to s' at time t



$$
\begin{aligned}
&\xi_{s,s'}(t)\\
&= P(Z_t = s, Z_{t+1} = s' \mid X_{1:T} = x_{1:T})\\
&\propto P(Z_t = s, Z_{t+1} = s', X_{1:T} = x_{1:T})\\
&= P(Z_t = s, X_{1:t} = x_{1:t}) P(Z_{t+1} = s', X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t} = x_{1:t})\\
&= \alpha_s(t) P(Z_{t+1} = s' \mid Z_t = s) P(X_{t+1:T} = x_{t+1:T} \mid Z_{t+1} = s')\\
&= \alpha_s(t) a_{s,s'} P(X_{t+1} = x_{t+1} \mid Z_{t+1} = s') P(X_{t+2:T} = x_{t+2:T} \mid Z_{t+1} = s')\\
&= \alpha_s(t) a_{s,s'} b_{s',x_{t+1}} \beta_{s'}(t+1)
\end{aligned}
$$

The normalization constant is in fact again $P(X_{1:T} = x_{1:T})$

Finding the most likely path

Can't use forward and backward messages directly to find the most likely path, but it is very similar to the forward procedure.

Finding the most likely path

Can't use forward and backward messages directly to find the most likely path, but it is very similar to the forward procedure. Key: compute

$$\delta_s(t) = \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t})$$

the probability of the most likely path for time $1:t$ ending at state s

Computing $\delta_s(t)$

Observe

$$\delta_s(t) = \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t})$$

Computing $\delta_s(t)$

Observe

$$\begin{aligned}\delta_s(t) &= \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) \\ &= \max_{s'} \max_{z_{1:t-2}} P(Z_t = s, Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t} = x_{1:t})\end{aligned}$$

Computing $\delta_s(t)$

Observe

$$\begin{aligned}
\delta_s(t) &= \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} \max_{z_{1:t-2}} P(Z_t = s, Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} P(Z_t = s \mid Z_{t-1} = s') P(X_t = x_t \mid Z_t = s) \cdot \\
&\qquad \max_{z_{1:t-2}} P(Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t-1} = x_{1:t-1})
\end{aligned}$$

Computing $\delta_s(t)$

Observe

$$
\begin{aligned}
\delta_s(t) &= \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} \max_{z_{1:t-2}} P(Z_t = s, Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} P(Z_t = s \mid Z_{t-1} = s') P(X_t = x_t \mid Z_t = s) \cdot \\
&\qquad\qquad \max_{z_{1:t-2}} P(Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t-1} = x_{1:t-1}) \\
&= b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1) \qquad \text{(recursion!)}
\end{aligned}
$$

Computing $\delta_s(t)$

Observe

$$\begin{aligned}
\delta_s(t) &= \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} \max_{z_{1:t-2}} P(Z_t = s, Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} P(Z_t = s \mid Z_{t-1} = s') P(X_t = x_t \mid Z_t = s) \cdot \\
&\qquad \max_{z_{1:t-2}} P(Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t-1} = x_{1:t-1}) \\
&= b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1) && \text{(recursion!)}
\end{aligned}$$

Base case: $\delta_s(1) = P(Z_1 = s, X_1 = x_1) = \pi_s b_{s,x_1}$

Computing $\delta_s(t)$

Observe

$$
\begin{aligned}
\delta_s(t) &= \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} \max_{z_{1:t-2}} P(Z_t = s, Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t} = x_{1:t}) \\
&= \max_{s'} P(Z_t = s \mid Z_{t-1} = s') P(X_t = x_t \mid Z_t = s) \cdot \\
&\qquad \max_{z_{1:t-2}} P(Z_{t-1} = s', Z_{1:t-2} = z_{1:t-2}, X_{1:t-1} = x_{1:t-1}) \\
&= b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1) \qquad \text{(recursion!)}
\end{aligned}
$$

Base case: $\delta_s(1) = P(Z_1 = s, X_1 = x_1) = \pi_s b_{s,x_1}$

Exactly the same as forward messages except replacing "sum" by "max"!

Viterbi Algorithm (!)

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

Viterbi Algorithm (!)

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

For each $t = 2, \ldots, T$,

- for each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1),$$

Viterbi Algorithm (!)

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

For each $t = 2, \ldots, T$,

- for each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1),$$
$$\Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s} \delta_{s'}(t-1).$$

Viterbi Algorithm (!)

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

For each $t = 2, \ldots, T$,

- for each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1),$$
$$\Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s} \delta_{s'}(t-1).$$

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

Viterbi Algorithm (!)

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

For each $t = 2, \ldots, T$,

- for each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1),$$
$$\Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s} \delta_{s'}(t-1).$$

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.
For each $t = T, \ldots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$.

Viterbi Algorithm (!)

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

For each $t = 2, \ldots, T$,

- for each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1),$$
$$\Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s} \delta_{s'}(t-1).$$

Backtracking: let $z_T^* = \mathrm{argmax}_s\, \delta_s(T)$.
For each $t = T, \ldots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$.

Output the most likely path $z_1^*, \ldots, z_T^*$.

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

...

Example: $x_{1:4} =$ clean, shop, walk, walk

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

. . .

$$\delta_{\text{sunny}}(1) = \pi_{\text{sunny}} b_{\text{sunny},\text{clean}}$$

Example: $x_{1:4} =$ clean, shop, walk, walk

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

$\ldots$

$$\delta_{\text{sunny}}(1) = \pi_{\text{sunny}} b_{\text{sunny},\text{clean}}$$

$$\delta_{\text{rainy}}(1) = \pi_{\text{rainy}} b_{\text{rainy},\text{clean}}$$

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

For each $s \in [S]$, compute $\delta_s(1) = \pi_s b_{s,x_1}$.

...

$$\delta_{\text{sunny}}(1) = \pi_{\text{sunny}} b_{\text{sunny},\text{clean}}$$

$$\delta_{\text{rainy}}(1) = \pi_{\text{rainy}} b_{\text{rainy},\text{clean}}$$

$$\boxed{\delta_{\text{sunny}}(1) = 0.25}$$

$$\boxed{\delta_{\text{rainy}}(1) = 0.4}$$

(numbers are made up here)

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$\boxed{\delta_{\text{sunny}}(1) = 0.25}$

$\boxed{\delta_{\text{rainy}}(1) = 0.4}$

Example: $x_{1:4} =$ clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(2) = b_{\text{sunny,shop}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{sunny}}\delta_{s'}(1)$$

$\boxed{\delta_{\text{sunny}}(1) = 0.25}$

$\boxed{\delta_{\text{rainy}}(1) = 0.4}$

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(2) = b_{\text{sunny},\text{shop}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{sunny}}\delta_{s'}(1)$$

$$\delta_{\text{rainy}}(2) = b_{\text{rainy},\text{shop}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{rainy}}\delta_{s'}(1)$$

$\boxed{\delta_{\text{sunny}}(1) = 0.25}$

$\boxed{\delta_{\text{rainy}}(1) = 0.4}$

Example: $x_{1:4} =$ clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(2) = b_{\text{sunny,shop}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{sunny}}\delta_{s'}(1)$$

$$\delta_{\text{rainy}}(2) = b_{\text{rainy,shop}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{rainy}}\delta_{s'}(1)$$

$\boxed{\delta_{\text{sunny}}(1) = 0.25}$ $\boxed{\delta_{\text{sunny}}(2) = 0.1}$

$\boxed{\delta_{\text{rainy}}(1) = 0.4}$ $\boxed{\delta_{\text{rainy}}(2) = 0.19}$

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(2) = b_{\text{sunny,shop}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{sunny}}\delta_{s'}(1)$$

$$\delta_{\text{rainy}}(2) = b_{\text{rainy,shop}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{rainy}}\delta_{s'}(1)$$

$$\begin{tikzcd}
\boxed{\delta_{\text{sunny}}(1) = 0.25} & \boxed{\delta_{\text{sunny}}(2) = 0.1} \arrow[l] \\
\boxed{\delta_{\text{rainy}}(1) = 0.4} & \boxed{\delta_{\text{rainy}}(2) = 0.19} \arrow[l]
\end{tikzcd}$$

Arrows represent the "argmax", i.e. $\Delta_s(t)$.

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(3) = b_{\text{sunny,walk}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{sunny}}\delta_{s'}(2)$$

$$\delta_{\text{rainy}}(3) = b_{\text{rainy,walk}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{rainy}}\delta_{s'}(2)$$

$$\begin{tikzcd}
\boxed{\delta_{\text{sunny}}(1) = 0.25} & \boxed{\delta_{\text{sunny}}(2) = 0.1} \arrow[l] \\
\boxed{\delta_{\text{rainy}}(1) = 0.4} & \boxed{\delta_{\text{rainy}}(2) = 0.19} \arrow[l]
\end{tikzcd}$$

Arrows represent the "argmax", i.e. $\Delta_s(t)$.

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(3) = b_{\text{sunny},\text{walk}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{sunny}}\delta_{s'}(2)$$

$$\delta_{\text{rainy}}(3) = b_{\text{rainy},\text{walk}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{rainy}}\delta_{s'}(2)$$

$$\begin{tikzcd}
\boxed{\delta_{\text{sunny}}(1) = 0.25} & \boxed{\delta_{\text{sunny}}(2) = 0.1} \arrow[l] & \boxed{\delta_{\text{sunny}}(3) = 0.04} \\
\boxed{\delta_{\text{rainy}}(1) = 0.4} & \boxed{\delta_{\text{rainy}}(2) = 0.19} \arrow[l] & \boxed{\delta_{\text{rainy}}(3) = 0.042}
\end{tikzcd}$$

Arrows represent the "argmax", i.e. $\Delta_s(t)$.

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(3) = b_{\text{sunny,walk}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{sunny}}\delta_{s'}(2)$$

$$\delta_{\text{rainy}}(3) = b_{\text{rainy,walk}} \max_{s' \in \{\text{sunny},\text{rainy}\}} a_{s',\text{rainy}}\delta_{s'}(2)$$

$$\begin{tikzcd}
\boxed{\delta_{\text{sunny}}(1) = 0.25} & \boxed{\delta_{\text{sunny}}(2) = 0.1} \arrow[l] & \boxed{\delta_{\text{sunny}}(3) = 0.04} \arrow[dl] \\
\boxed{\delta_{\text{rainy}}(1) = 0.4} & \boxed{\delta_{\text{rainy}}(2) = 0.19} \arrow[l] & \boxed{\delta_{\text{rainy}}(3) = 0.042} \arrow[ul]
\end{tikzcd}$$

Arrows represent the "argmax", i.e. $\Delta_s(t)$.

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(4) = b_{\text{sunny,walk}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{sunny}}\delta_{s'}(3)$$

$$\delta_{\text{rainy}}(4) = b_{\text{rainy,walk}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{rainy}}\delta_{s'}(3)$$

$$\begin{tikzcd}
\boxed{\delta_{\text{sunny}}(1) = 0.25} & \boxed{\delta_{\text{sunny}}(2) = 0.1} \arrow[l] & \boxed{\delta_{\text{sunny}}(3) = 0.04} \arrow[dl] & \boxed{\delta_{\text{sunny}}(4) = 0.016} \\
\boxed{\delta_{\text{rainy}}(1) = 0.4} & \boxed{\delta_{\text{rainy}}(2) = 0.19} \arrow[l] & \boxed{\delta_{\text{rainy}}(3) = 0.042} \arrow[ul] & \boxed{\delta_{\text{rainy}}(4) = 0.01}
\end{tikzcd}$$

Arrows represent the "argmax", i.e. $\Delta_s(t)$.

Example: $x_{1:4} =$ clean, shop, walk, walk

Viterbi Algorithm

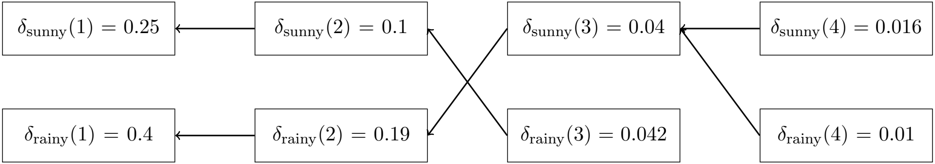
...

For each $t = 2, \ldots, T$ and each $s \in [S]$, compute

$$\delta_s(t) = b_{s,x_t} \max_{s'} a_{s',s}\delta_{s'}(t-1), \quad \Delta_s(t) = \operatorname*{argmax}_{s'} a_{s',s}\delta_{s'}(t-1).$$

$$\delta_{\text{sunny}}(4) = b_{\text{sunny,walk}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{sunny}}\delta_{s'}(3)$$

$$\delta_{\text{rainy}}(4) = b_{\text{rainy,walk}} \max_{s' \in \{\text{sunny,rainy}\}} a_{s',\text{rainy}}\delta_{s'}(3)$$



Arrows represent the "argmax", i.e. $\Delta_s(t)$.

Example: $x_{1:4}$ = clean, shop, walk, walk

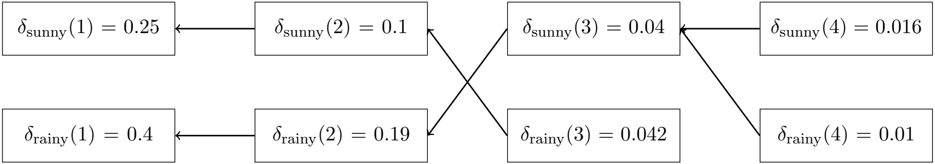
Viterbi Algorithm

...

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \ldots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$.

Output the most likely path $z_1^*, \ldots, z_T^*$.



The most likely path is “ ”

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

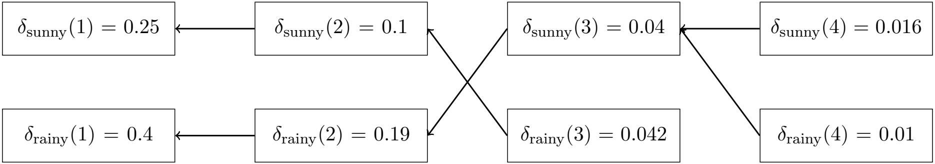
...

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \dots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$.

Output the most likely path $z_1^*, \dots, z_T^*$.

$$z_4^* = \operatorname*{argmax}_{s \in \{\text{sunny}, \text{rainy}\}} \delta_s(4)$$



The most likely path is “ ”

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

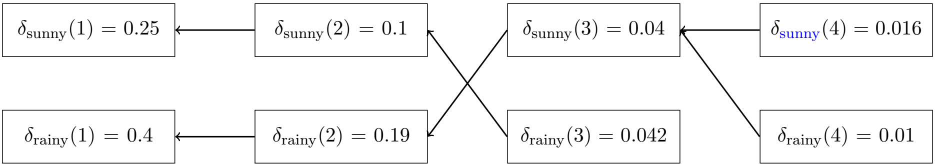
...

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \ldots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$.

Output the most likely path $z_1^*, \ldots, z_T^*$.

$$z_4^* = \underset{s \in \{\text{sunny}, \text{rainy}\}}{\operatorname{argmax}} \delta_s(4) = \text{sunny},$$



The most likely path is "**sunny**"

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

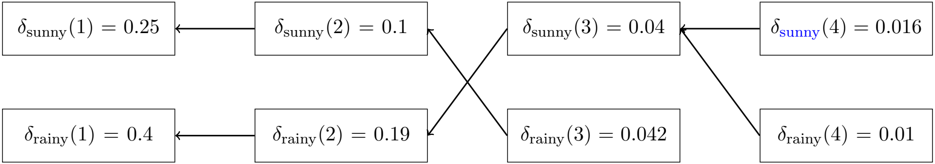
$\ldots$

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \ldots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$.

Output the most likely path $z_1^*, \ldots, z_T^*$.

$$z_4^* = \operatorname*{argmax}_{s \in \{\text{sunny}, \text{rainy}\}} \delta_s(4) = \text{sunny}, \quad z_3^* = \Delta_{\text{sunny}}(4)$$



The most likely path is "**sunny**"

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

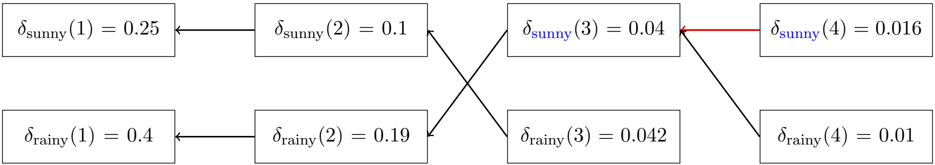
...

Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \dots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$. (just follow the arrow!)

Output the most likely path $z_1^*, \dots, z_T^*$.

$$z_4^* = \operatorname*{argmax}_{s \in \{\text{sunny},\text{rainy}\}} \delta_s(4) = \text{sunny}, \quad z_3^* = \Delta_{\text{sunny}}(4) = \text{sunny}$$



The most likely path is **“ sunny, sunny”**

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

. . .

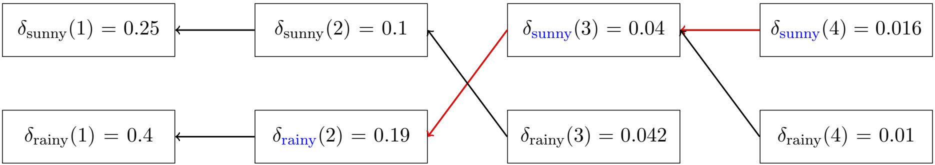
Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \ldots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$. (just follow the arrow!)

Output the most likely path $z_1^*, \ldots, z_T^*$.

$$z_4^* = \operatorname*{argmax}_{s \in \{\text{sunny},\text{rainy}\}} \delta_s(4) = \text{sunny}, \quad z_3^* = \Delta_{\text{sunny}}(4) = \text{sunny}$$

$$z_2^* = \Delta_{\text{sunny}}(3) = \text{rainy},$$



The most likely path is "**rainy, sunny, sunny**"

Example: $x_{1:4}$ = clean, shop, walk, walk

Viterbi Algorithm

...

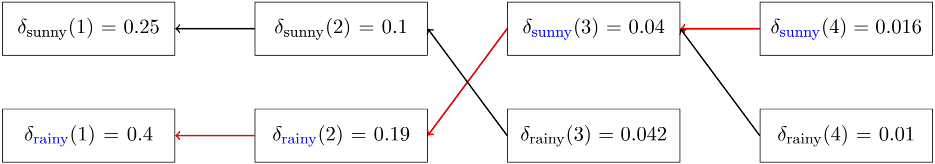
Backtracking: let $z_T^* = \operatorname{argmax}_s \delta_s(T)$.

For each $t = T, \dots, 2$: set $z_{t-1}^* = \Delta_{z_t^*}(t)$. (just follow the arrow!)

Output the most likely path $z_1^*, \dots, z_T^*$.

$$z_4^* = \operatorname*{argmax}_{s \in \{\text{sunny}, \text{rainy}\}} \delta_s(4) = \text{sunny}, \quad z_3^* = \Delta_{\text{sunny}}(4) = \text{sunny}$$

$$z_2^* = \Delta_{\text{sunny}}(3) = \text{rainy}, \qquad z_1^* = \Delta_{\text{rainy}}(2) = \text{rainy}$$



The most likely path is **"rainy, rainy, sunny, sunny"**

Learning the parameters of an HMM

All previous inferences depend on knowing the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.

Learning the parameters of an HMM

All previous inferences depend on knowing the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.

How do we learn the parameters based on N observation sequences $x_{n,1}, \ldots, x_{n,T}$ for $n = 1, \ldots, N$?

Learning the parameters of an HMM

All previous inferences depend on knowing the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.

How do we learn the parameters based on N observation sequences $x_{n,1}, \ldots, x_{n,T}$ for $n = 1, \ldots, N$?

MLE is intractable due to the hidden variables $Z_{n,t}$'s (similar to GMMs)

Learning the parameters of an HMM

All previous inferences depend on knowing the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.

How do we learn the parameters based on N observation sequences $x_{n,1}, \ldots, x_{n,T}$ for $n = 1, \ldots, N$?

MLE is intractable due to the hidden variables $Z_{n,t}$'s (similar to GMMs)

Need to apply **EM** again! Known as the **Baum–Welch algorithm**.

Applying EM: E-Step

Recall in the E-Step we fix the parameters and find the **posterior distributions q of the hidden states** (for each sample n),

Applying EM: E-Step

Recall in the E-Step we fix the parameters and find the **posterior distributions q of the hidden states** (for each sample n), which leads to the complete log-likelihood:

$$\mathbb{E}_{z_{1:T} \sim q} \left[\ln P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) \right]$$

Applying EM: E-Step

Recall in the E-Step we fix the parameters and find the **posterior distributions q of the hidden states** (for each sample n), which leads to the complete log-likelihood:

$$\mathbb{E}_{z_{1:T}\sim q}\left[\ln P(Z_{1:T}=z_{1:T}, X_{1:T}=x_{1:T})\right]$$
$$=\mathbb{E}_{z_{1:T}\sim q}\left[\ln \pi_{z_1} + \sum_{t=1}^{T-1}\ln a_{z_t,z_{t+1}} + \sum_{t=1}^{T}\ln b_{z_t,x_t}\right]$$

Applying EM: E-Step

Recall in the E-Step we fix the parameters and find the **posterior distributions q of the hidden states** (for each sample n), which leads to the complete log-likelihood:

$$
\begin{aligned}
&\mathbb{E}_{z_{1:T}\sim q}\left[\ln P(Z_{1:T}=z_{1:T}, X_{1:T}=x_{1:T})\right] \\
&= \mathbb{E}_{z_{1:T}\sim q}\left[\ln \pi_{z_1} + \sum_{t=1}^{T-1} \ln a_{z_t,z_{t+1}} + \sum_{t=1}^{T} \ln b_{z_t,x_t}\right] \\
&= \sum_s \gamma_s(1)\ln \pi_s + \sum_{t=1}^{T-1}\sum_{s,s'} \xi_{s,s'}(t)\ln a_{s,s'} + \sum_{t=1}^{T}\sum_s \gamma_s(t)\ln b_{s,x_t}
\end{aligned}
$$

We have discussed how to compute

$$
\begin{aligned}
\gamma_s(t) &= P(Z_t = s \mid X_{1:T}=x_{1:T}) \\
\xi_{s,s'}(t) &= P(Z_t = s, Z_{t+1}=s' \mid X_{1:T}=x_{1:T})
\end{aligned}
$$

Applying EM: M-Step

The maximizer of complete log-likelihood is simply doing **weighted counting** (compared to the unweighted counting on Slide 36):

$$\pi_s \propto \sum_n \gamma_s^{(n)}(1) = \mathbb{E}_q\left[\ \textbf{\#initial states with value } s\right]$$

$$a_{s,s'} \propto \sum_n \sum_{t=1}^{T-1} \xi_{s,s'}^{(n)}(t) = \mathbb{E}_q\left[\ \textbf{\#transitions from } s \textbf{ to } s'\right]$$

$$b_{s,o} \propto \sum_n \sum_{t:x_t=o} \gamma_s^{(n)}(t) = \mathbb{E}_q\left[\ \textbf{\#state-outcome pairs } (s,o)\right]$$

Applying EM: M-Step

The maximizer of complete log-likelihood is simply doing **weighted counting** (compared to the unweighted counting on Slide 36):

$$\pi_s \propto \sum_n \gamma_s^{(n)}(1) = \mathbb{E}_q\left[\ \textbf{\#initial states with value } s\right]$$

$$a_{s,s'} \propto \sum_n \sum_{t=1}^{T-1} \xi_{s,s'}^{(n)}(t) = \mathbb{E}_q\left[\ \textbf{\#transitions from } s \textbf{ to } s'\right]$$

$$b_{s,o} \propto \sum_n \sum_{t:x_t=o} \gamma_s^{(n)}(t) = \mathbb{E}_q\left[\ \textbf{\#state-outcome pairs } (s,o)\right]$$

where

$$\gamma_s^{(n)}(t) = P(Z_{n,t} = s \mid X_{n,1:T} = x_{n,1:T})$$

$$\xi_{s,s'}^{(n)}(t) = P(Z_{n,t} = s, Z_{n,t+1} = s' \mid X_{n,1:T} = x_{n,1:T})$$

Applying EM: M-Step

The maximizer of complete log-likelihood is simply doing **weighted counting** (compared to the unweighted counting on Slide 36):

$$\pi_s \propto \sum_n \gamma_s^{(n)}(1) = \mathbb{E}_q \left[\text{ \textbf{\#initial states with value} } s \right]$$

$$a_{s,s'} \propto \sum_n \sum_{t=1}^{T-1} \xi_{s,s'}^{(n)}(t) = \mathbb{E}_q \left[\text{ \textbf{\#transitions from} } s \text{ \textbf{to} } s' \right]$$

$$b_{s,o} \propto \sum_n \sum_{t:x_t=o} \gamma_s^{(n)}(t) = \mathbb{E}_q \left[\text{ \textbf{\#state-outcome pairs} } (s,o) \right]$$

where

$$\gamma_s^{(n)}(t) = P(Z_{n,t} = s \mid X_{n,1:T} = x_{n,1:T})$$
$$\xi_{s,s'}^{(n)}(t) = P(Z_{n,t} = s, Z_{n,t+1} = s' \mid X_{n,1:T} = x_{n,1:T})$$

(Recall how EM for GMM updates parameters using weighted averages.)

Baum–Welch algorithm

Step 0 Initialize the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

Baum–Welch algorithm

Step 0 Initialize the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

Step 1 (E-Step) Fixing the parameters, compute forward and backward messages for all sample sequences, then use these to compute $\gamma_s^{(n)}(t)$ and $\xi_{s,s'}^{(n)}(t)$ for each n, t, s, s' (see Slides 47 and 48).

Baum–Welch algorithm

Step 0 Initialize the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

Step 1 (E-Step) Fixing the parameters, compute forward and backward messages for all sample sequences, then use these to compute $\gamma_s^{(n)}(t)$ and $\xi_{s,s'}^{(n)}(t)$ for each n, t, s, s' (see Slides 47 and 48).

Step 2 (M-Step) Update parameters:

$$\pi_s \propto \sum_n \gamma_s^{(n)}(1), \quad a_{s,s'} \propto \sum_n \sum_{t=1}^{T-1} \xi_{s,s'}^{(n)}(t), \quad b_{s,o} \propto \sum_n \sum_{t:x_t=o} \gamma_s^{(n)}(t)$$

Baum–Welch algorithm

Step 0 Initialize the parameters $(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

Step 1 (E-Step) Fixing the parameters, compute forward and backward messages for all sample sequences, then use these to compute $\gamma_s^{(n)}(t)$ and $\xi_{s,s'}^{(n)}(t)$ for each n, t, s, s' (see Slides 47 and 48).

Step 2 (M-Step) Update parameters:

$$\pi_s \propto \sum_n \gamma_s^{(n)}(1), \quad a_{s,s'} \propto \sum_n \sum_{t=1}^{T-1} \xi_{s,s'}^{(n)}(t), \quad b_{s,o} \propto \sum_n \sum_{t:x_t=o} \gamma_s^{(n)}(t)$$

Step 3 Return to Step 1 if not converged

Summary

Very important models: **Markov chains**, **hidden Markov models**

Summary

Very important models: **Markov chains**, **hidden Markov models**

Several algorithms:

- forward and backward procedures
- inferring HMMs based on forward and backward messages
- Viterbi algorithm and variants (see today's discussion)
- Baum–Welch algorithm