



École Polytechnique de Louvain

LINFO2275 - Data Mining and Decision Making

Examen Questions

June 17, 2025

Mathis DELSART - 31302100

Contents

1 Question 1 - PCA	1
2 Question 2 - DA	5
3 Question 3 - CCA	9
4 Question 4 - MCA	13
5 Question 5 - MDS	17
6 Question 6 - Edit Distance (DP)	21
7 Question 7 - Bellman-Ford (DP)	25
8 Question 8 - DTW	29
9 Question 9 et 10 - MDP / QLearning	33
10 Question 11 - Information Retrieval	37
11 Question 12 - Information Retrieval	39
12 Question 13 - PageRank	43
13 Question 14 - PageRank	47
14 Question 15 - Collab. Recom. Model	51
15 Question 16 - Collab. Recom. Model	53
16 Question 17 - Collab. Recom. Model	56
17 Question 18 - Markov Chain (Evol. Eq.)	60
18 Question 19 - Markov Chain (# Of Visits)	62
19 Question 20 - Markov Chain (Absorb. Probab.)	66
20 Question 21 - Markov Chain (Lifetime Value)	70
21 Question 22 - HMM & Likelihood	72
22 Question 23 - Fair Prediction in SC	76

Question 1 - PCA

Describe and derive in detail the method allowing to obtain the principal axes of a principal components analysis (one technique based on the empirical data and one technique based on the mathematical definitions). Interpret the results, e.g., what is the interpretation of the eigenvalues? How can we obtain the coordinates of the data in the principal components system? Finally, describe some methods for performing feature selection (and not extraction).

Describe and derive in detail the method allowing to obtain the principal axes of a principal components analysis (one technique based on the empirical data and one technique based on the mathematical definitions).

PCA (Analyse en Composantes Principales) est une technique linéaire d'extraction de caractéristiques utilisée pour l'exploration et la visualisation de données numériques. Elle projette les données dans un espace de dimension réduite tout en conservant un maximum de variance, afin de préserver l'information essentielle. Les nouvelles dimensions, appelées composantes principales, sont des axes orthogonaux successifs qui capturent chacun la plus grande variance restante. Le nombre de composantes est déterminé par l'utilisateur.

On définit g , le centroïde du nuage de points, comme la moyenne des observations : $g = \frac{1}{n} \sum_{i=1}^n x_i$ où x_i désigne le vecteur des variables pour l'observation i , et n le nombre total d'observations.

Technique based on the empirical data

Le but de la PCA est de déterminer la direction v (vecteur unitaire) qui maximise la variance des données projetées. La variance des données initiales s'exprime comme :

$$\begin{aligned}\sigma_v^2 &= \frac{1}{2n(n-1)} \sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^T (x_i - x_j) \\ &= \frac{1}{2n(n-1)} \sum_{i=1}^n \sum_{j=1}^n ((x_i - g) + (g - x_j))^T ((x_i - g) + (g - x_j)) \\ &= \frac{1}{2n(n-1)} \sum_{i=1}^n \sum_{j=1}^n ((x_i - g)^T (x_i - g) + (g - x_j)^T (g - x_j) + 2(x_i - g)^T (g - x_j)) \\ &= \frac{1}{n-1} \sum_{i=1}^n (x_i - g)^T (x_i - g)\end{aligned}$$

la variance des données projetée sur l'axe défini par v s'exprime comme :

$$\sigma_v^2 = \frac{1}{n-1} \sum_{i=1}^n \left((\pi(x_i - g))^T (\pi(x_i - g)) \right)$$

où $\pi = vv^T$ est l'opérateur de projection orthogonale sur l'axe dirigé par v , et g est le centroïde du nuage de points. Le vecteur v est unitaire, c'est-à-dire $v^T v = 1$. Il est essentiel de centrer les données (i.e., soustraire g à chaque observation) pour éviter un biais dû à la position du nuage de points dans l'espace.

Détaillons le développement de cette expression pour faire apparaître la matrice de covariance.

$$\begin{aligned}\sigma_v^2 &= \frac{1}{n-1} \sum_{i=1}^n (\pi(x_i - g))^T (\pi(x_i - g)) \\ &= \frac{1}{n-1} \sum_{i=1}^n (vv^T(x_i - g))^T (vv^T(x_i - g)) \\ &= \frac{1}{n-1} \sum_{i=1}^n (x_i - g)^T (vv^T)^T (vv^T)(x_i - g)\end{aligned}$$

Vu que $v^T v = 1$, on peut écrire :

$$\begin{aligned}
\sigma_v^2 &= \frac{1}{n-1} \sum_{i=1}^n (x_i - g)^T v v^T (x_i - g) \\
&= \frac{1}{n-1} \sum_{i=1}^n ((x_i - g)^T v) (v^T (x_i - g)) \\
&= \frac{1}{n-1} \sum_{i=1}^n v^T (x_i - g) (x_i - g)^T v \\
&= v^T \left[\frac{1}{n-1} \sum_{i=1}^n (x_i - g) (x_i - g)^T \right] v \\
&= v^T \Sigma v
\end{aligned}$$

où Σ est la matrice de variance-covariance des données (mesure de la covariance entre les features). Cette matrice est symétrique et définie positive.

Le but est de maximiser la quantité suivante sous la contrainte de normalisation :

$$\max_v \{v^T \Sigma v\} \quad \text{sous la contrainte} \quad v^T v = 1$$

Pour déterminer le second axe principal, on impose en plus une contrainte d'orthogonalité au premier vecteur propre v_1 : $v^T v_1 = 0$.

Pour résoudre ce problème, on utilise la méthode des multiplicateurs de Lagrange. On définit la fonction Lagrangienne :

$$\mathcal{L}(v, \lambda) = v^T \Sigma v + \lambda(1 - v^T v)$$

où λ est le multiplicateur de Lagrange.

En dérivant \mathcal{L} par rapport à v et en annulant le gradient, on obtient :

$$\frac{\partial \mathcal{L}}{\partial v} = 2\Sigma v - 2\lambda v = 0$$

ce qui revient à résoudre le problème aux valeurs propres :

$$\Sigma v = \lambda v$$

Puisque Σ est définie positive, toutes ses valeurs propres sont strictement positives ($\lambda_i > 0 \quad \forall i$) et ses vecteurs propres associés peuvent être choisis orthogonaux deux à deux. Les axes définis par ces vecteurs propres v sont appelés composantes principales.

La variance des données projetées sur un vecteur v s'exprime donc par :

$$\sigma_v^2 = v^T \Sigma v = v^T \lambda v = \lambda v^T v = \lambda,$$

puisque v est normalisé à 1 et que $\Sigma v = \lambda v$.

Ainsi, la solution consiste à projeter les données sur les vecteurs propres de Σ associés aux plus grandes valeurs propres, car ces directions maximisent la variance projetée.

Technique basée sur les définitions mathématiques

Soit $x = [x_1, x_2, \dots, x_p]^T$ un vecteur aléatoire contenant p variables aléatoires correspondant aux caractéristiques observées. On définit :

$$y = v^T x$$

avec la contrainte $v^T v = 1$, où v est un vecteur unitaire qui sert de vecteur directeur pour la projection de x .

L'espérance de y s'écrit alors :

$$\mathbb{E}[y] = \mathbb{E}[v^T x] = v^T \mathbb{E}[x] = v^T g$$

où $g = \mathbb{E}[x]$ est le vecteur des moyennes.

La variance de la variable projetée y s'écrit :

$$\sigma_y^2 = \mathbb{E}[(y - \mathbb{E}[y])^2] = \mathbb{E}[(v^T x - v^T g)^2]$$

En développant, cela donne :

$$\sigma_y^2 = \mathbb{E}[v^T(x - g)(x - g)^T v] = v^T \mathbb{E}[(x - g)(x - g)^T] v = v^T S v$$

où S est la matrice de covariance des données.

L'objectif est donc de maximiser la variance projetée, soit :

$$\max_v \{v^T S v\} \quad \text{sous la contrainte} \quad v^T v = 1$$

La matrice S peut être estimée à partir des données observées $\{x_i\}_{i=1}^n$ par :

$$S \approx \hat{S} = \frac{1}{n-1} \sum_{i=1}^n (x_i - g)(x_i - g)^T \approx \Sigma$$

ce qui nous ramène à la même formulation que l'approche précédente.

Interpret the results, e.g., what is the interpretation of the eigenvalues?

Dans le contexte de l'analyse en composantes principales (PCA), chaque valeur propre λ_i correspond à la variance des données projetées sur le vecteur propre v_i associé. Autrement dit, chaque valeur propre mesure la quantité de variance expliquée par la composante principale correspondante.

La somme de toutes les valeurs propres représente la variance totale des données :

$$\text{tr}(\Sigma) = \sum_{i=1}^n \lambda_i = \sigma^2,$$

où Σ est la matrice de covariance des données et $\text{tr}(\cdot)$ désigne la trace d'une matrice.

Il est important de standardiser les features avant d'appliquer une PCA, afin que chacune contribue équitablement à l'analyse. En effet, sans standardisation, les variables ayant une plus grande variance (généralement dues à leur échelle) domineraient la construction des composantes principales, faussant ainsi les résultats. La standardisation permet donc de rendre les variances comparables, en les ramenant à une échelle commune.

Le pourcentage de variance expliquée par les k premières composantes principales est donné par :

$$\text{Variance expliquée} = \frac{\sum_{j=1}^k \lambda_j}{\sum_{i=1}^n \lambda_i} = \frac{\sum_{j=1}^k \lambda_j}{\sigma_v^2}.$$

Ce ratio permet de choisir un nombre réduit de composantes principales tout en conservant une grande partie de l'information présente dans les données initiales.

How can we obtain the coordinates of the data in the principal components system?

Pour obtenir les coordonnées des données dans le nouveau système formé par les composantes principales, il suffit de projeter les vecteurs de données sur les vecteurs propres associés aux plus grandes valeurs propres.

Soit $x \in \mathbb{R}^p$ un vecteur de données centré (i.e., $x - g$, où $g = \mathbb{E}[x]$ ou la moyenne empirique), et soit $V = [v_1, v_2, \dots, v_k] \in \mathbb{R}^{p \times k}$ la matrice formée des k premiers vecteurs propres (orthonormés).

Les coordonnées de x dans le nouveau repère sont alors données par la projection :

$$x_{proj} = V^T x.$$

La projection d'un vecteur individuel x sur un axe principal v correspond à la quantité :

$$v^T x,$$

ce qui représente à la fois :

- la longueur de la projection de x sur v ,
- et la coordonnée de x dans la direction de cette composante principale.

Ainsi, chaque valeur $v_i^\top x$ mesure combien x est aligné avec la i -ème composante principale.

Finally, describe some methods for performing feature selection (and not extraction).

Les méthodes de feature selection, contrairement aux méthodes de feature extraction, ne projettent pas les données dans un espace de dimension réduite et ne transforment donc pas les variables d'origine. Elles consistent plutôt à éliminer les variables les moins pertinentes pour la tâche d'apprentissage.

Ces méthodes sont utilisées pour :

- Lutter contre la malédiction de la dimensionnalité (curse of dimensionality)
- Améliorer la généralisation des modèles en réduisant le risque de surapprentissage (overfitting)
- accélérer le processus d'entraînement
- améliorer l'interprétabilité des modèles

Il existe plusieurs approches principales :

- **Maximum Relevance Selection** : cette méthode calcule un score mesurant la pertinence de chaque variable par rapport à la variable cible. On sélectionne ensuite les variables ayant les scores les plus élevés. Ces scores peuvent être basés sur la corrélation, l'information mutuelle, des tests statistiques, etc.
- **Minimal Redundancy Selection** : cette approche vise à éliminer les variables qui sont fortement corrélées entre elles. En effet, deux variables très similaires n'apportent pas d'information complémentaire utile.
- **mRMR (minimum Redundancy Maximum Relevance)** : il s'agit d'une combinaison des deux critères précédents : on cherche à sélectionner les variables à la fois fortement liées à la cible et peu redondantes entre elles.
- **Stepwise Regression / Selection** : méthode gloutonne et itérative qui ajoute ou retire des variables en fonction de leur contribution à la performance du modèle. Le critère de sélection peut être basé sur un test de vraisemblance (likelihood ratio test), avec arrêt lorsque l'ajout ou le retrait d'une variable ne génère plus d'amélioration significative.
- **L1 Regularization (Lasso)** : cette méthode repose sur l'ajout d'un terme de pénalisation proportionnel à la norme L1 des coefficients dans la fonction de perte. Elle pousse certains coefficients à exactement zéro, réalisant ainsi une sélection implicite des variables. L'avantage est qu'elle s'effectue directement pendant l'entraînement du modèle, sans nécessiter de prétraitement.
- **Méthodes embarquées** : certains modèles réalisent automatiquement une sélection de variables, comme les arbres de décision ou les méthodes ensemblistes comme le bagging (e.g., Random Forest), qui attribuent des importances aux variables en fonction de leur utilité dans les décisions prises.

Question 2 - DA

Derive and explain the decomposition of the variance in within- and between-cluster variance as well as the technique allowing to obtain the factorial axes of a discriminant analysis. Interpret these results – how can we interpret the computed eigenvalues? In addition, explain the intuition behind the classification algorithm that is used in the context of a discriminant analysis after having projected the data on the discriminant axes (a mixture of gaussians). Finally, describe some methods for performing feature selection (and not extraction).

Derive and explain the decomposition of the variance in within- and between-cluster variance as well as the technique allowing to obtain the factorial axes of a discriminant analysis.

L'analyse discriminante est une méthode de réduction de dimension supervisée (feature extraction) qui, contrairement à la PCA, tient compte de l'étiquetage des données. Elle est également utilisée comme algorithme de classification. Cette méthode fonctionne sur les données numériques mais peut être étendue à des données catégorielles. Son objectif est de trouver une projection des données dans un espace de plus faible dimension qui maximise la séparabilité entre les classes.

Mathématiquement, on cherche une projection linéaire qui maximise le rapport entre la variance inter-classe et la variance totale des données projetées.

Soit g le centroïde global des données. La variance totale est donnée par :

$$\begin{aligned}\sigma^2 &= \frac{1}{2n^2} \sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^T (x_i - x_j) \\ &= \frac{1}{n} \sum_{i=1}^n (x_i - g)^T (x_i - g), \quad \text{par définition de la variance} \\ &= \frac{1}{n} \sum_{k=1}^q \sum_{i \in C(k)} (x_i - g)^T (x_i - g) \\ &= \frac{1}{n} \sum_{k=1}^q SS(k)\end{aligned}$$

où q est le nombre de classes, $C(k)$ l'ensemble des observations appartenant à la classe k , et $SS(k)$ l'inertie (somme des carrés) pour la classe k .

Nous décomposons cette inertie en deux termes : la variance intra-classe (within-class) et la variance inter-classe (between-class). En développant $SS(k)$, on a :

$$\begin{aligned}SS(k) &= \sum_{i \in C(k)} (x_i - g)^T (x_i - g) \\ &= \sum_{i \in C(k)} [(x_i - g(k)) + (g(k) - g)]^T [(x_i - g(k)) + (g(k) - g)] \\ &= \sum_{i \in C(k)} (\|x_i - g(k)\|^2 + \|g(k) - g\|^2 + 2(x_i - g(k))^T (g(k) - g)) \\ &= \underbrace{\sum_{i \in C(k)} \|x_i - g(k)\|^2}_{SS_w(k)} + \underbrace{n(k)\|g(k) - g\|^2}_{SS_b(k)}, \quad \text{le terme croisé est nul car } \sum_{i \in C(k)} (x_i - g(k)) = 0\end{aligned}$$

On obtient donc :

- **Variance intra-classe** : $SS_w(k) = \sum_{i \in C(k)} \|x_i - g(k)\|^2$

C'est la somme des distances au carré entre les points d'un cluster $C(k)$ et son centre de gravité $g(k)$. Elle mesure la compacité du cluster : plus elle est faible, plus les points sont proches les uns des autres.

- **Variance inter-classe** : $SS_b(k) = n(k)\|g(k) - g\|^2$

C'est la distance au carré entre le centre du cluster $g(k)$ et le centre global g , pondérée par le nombre de points $n(k)$. Elle mesure la séparation entre les clusters : plus elle est grande, plus le cluster est éloigné des autres.

On en déduit la variance totale :

$$\begin{aligned}\sigma^2 &= \frac{1}{n} \sum_{k=1}^q (SS_w(k) + SS_b(k)) \\ &= \underbrace{\frac{1}{n} \sum_{k=1}^q \sum_{i \in C(k)} \|x_i - g(k)\|^2}_{\sigma_w^2 = \text{variance intra-classe}} + \underbrace{\frac{1}{n} \sum_{k=1}^q n(k) \|g(k) - g\|^2}_{\sigma_b^2 = \text{variance inter-classe}}\end{aligned}$$

Obtention des axes discriminants

Pour construire les axes factoriels, on cherche une projection $\pi(x) = v^T x$ (avec v un vecteur unitaire ($v^T v = 1$)) qui maximise le rapport :

$$J(v) = \frac{\sigma_{v(b)}^2}{\sigma_v^2} \in [0, 1]$$

où :

- $\sigma_{v(b)}^2 = v^T B v$, avec B la matrice de covariance inter-classes,

$$\begin{aligned}\sigma_{v(b)}^2 &= \frac{1}{n} \sum_{k=1}^q n(k) (\pi g(k) - \pi g)^T (\pi g(k) - \pi g) \\ &= v^T \left(\sum_{k=1}^q n(k) \frac{(g(k) - g)(g(k) - g)^T}{n} \right) v \\ &= v^T B v\end{aligned}$$

- $\sigma_v^2 = v^T \Sigma v$, avec Σ la matrice de covariance totale.

$$\begin{aligned}\sigma_v^2 &= \frac{1}{n} \sum_{i=1}^n (\pi x_i - \pi g)^T (\pi x_i - \pi g) \\ &= v^T \left(\sum_{i=1}^n \frac{(x_i - g)(x_i - g)^T}{n} \right) v \\ &= v^T \Sigma v\end{aligned}$$

On souhaite donc maximiser le critère :

$$\max_v \{J(v)\} = \max_v \left\{ \frac{v^T B v}{v^T \Sigma v} \right\}$$

La solution de ce problème d'optimisation conduit à résoudre le problème généralisé aux valeurs propres :

$$\Sigma^{-1} B v = \lambda v$$

Les vecteurs propres v de cette équation donnent les directions des axes discriminants, et les valeurs propres $\lambda \in [0, 1]$ représentent l'importance relative de ces axes. Il y a au plus q_1 vecteurs propres non nuls, ce qui fixe la dimension maximale de l'espace discriminant. Cette limite est liée au rang de la matrice de variance inter-classe, qui est au plus q_1 . Cela signifie qu'on peut séparer au mieux q classes dans un espace de dimension q_1 , sans perte d'information discriminante.

On a obtenu ce résultat en résolvant l'équation suivante :

$$\partial_v \left(\frac{v^T B v}{v^T \Sigma v} \right) = 0$$

Ce qui donne le résultat :

$$2(v^T \Sigma v) B v - 2(v^T B v) \Sigma v = 0 \Leftrightarrow B v = \left(\frac{v^T B v}{v^T \Sigma v} \right) \Sigma v.$$

Avec λ , défini comme $\frac{v^T B v}{v^T \Sigma v}$.

Procédure :

1. Calculer la matrice $\Sigma^{-1}B$
2. Extraire les vecteurs propres associés aux plus grandes valeurs propres
3. Les normaliser : ils forment les directions des axes factoriels discriminants

Interpret these results – how can we interpret the computed eigenvalues?

Comme pour PCA, les vecteurs propres normalisés donnent les directions des nouveaux axes. La projection d'une observation x_i sur un axe discriminant est donnée par :

$$z_i = v^T(x_i - g)$$

Les valeurs propres associées mesurent le rapport de la variance expliquée entre les classes (inter-classe) sur la variance totale projetée. Ainsi, une valeur propre proche de 1 indique une forte séparabilité des classes le long de l'axe considéré. En pratique, on retient les axes ayant les plus grandes valeurs propres, car ils offrent la meilleure séparation entre les classes.

In addition, explain the intuition behind the classification algorithm that is used in the context of a discriminant analysis after having projected the data on the discriminant axes (a mixture of gaussians).

Après avoir projeté les données sur les axes discriminants (c'est-à-dire les directions maximisant la séparation inter-classes), la classification repose sur un modèle probabiliste, où l'on **suppose que les données projetées suivent une loi normale multivariée dans chaque classe**.

Plus formellement, on fait l'hypothèse que :

$$x \mid y = k \sim \mathcal{N}(\mu_k, \Sigma)$$

où :

- $y \in \{1, \dots, q\}$ est la variable de classe,
- μ_k est le centroïde (moyenne vectorielle) de la classe k ,
- Σ est la matrice de covariance commune à toutes les classes (hypothèse clé de LDA).

Le but de la classification est d'assigner un nouvel échantillon x à la classe k qui maximise la probabilité a posteriori :

$$P(y = k \mid x) \propto \underbrace{P(x \mid y = k)}_{\text{à posteriori}} \underbrace{P(y = k)}_{\text{à priori}}$$

En prenant le logarithme et en développant la densité gaussienne, on montre que la règle de décision est linéaire. On obtient une fonction discriminante linéaire δ_k pour chaque classe.

On assigne alors l'observation à la classe k pour laquelle $\delta_k(x)$ est maximale. Cette séparation est **linéaire** (hyperplan) car Σ est identique pour toutes les classes.

Intuition : même dans l'espace projeté, les observations de chaque classe forment des "nuages" gaussiens centrés autour de leur barycentre. Pour classifier une nouvelle observation, on ne regarde pas simplement sa distance euclidienne aux centres, mais sa **distance de Mahalanobis**, qui prend en compte la dispersion des données (via Σ).

Si on ne suppose plus que toutes les classes partagent la même matrice de covariance, alors les frontières ne sont plus linéaires mais quadratiques — on parle alors d'**analyse discriminante quadratique (QDA)**.

Note: Voir les Slides pour plus de détail.

Finally, describe some methods for performing feature selection (and not extraction).

Les méthodes de feature selection, contrairement aux méthodes de feature extraction, ne projettent pas les données dans un espace de dimension réduite et ne transforment donc pas les variables d'origine. Elles consistent plutôt à éliminer les variables les moins pertinentes pour la tâche d'apprentissage.

Ces méthodes sont utilisées pour :

- Lutter contre la malédiction de la dimensionnalité (curse of dimensionality)
- Améliorer la généralisation des modèles en réduisant le risque de surapprentissage (overfitting)
- accélérer le processus d'entraînement
- améliorer l'interprétabilité des modèles

Il existe plusieurs approches principales :

- **Maximum Relevance Selection** : cette méthode calcule un score mesurant la pertinence de chaque variable par rapport à la variable cible. On sélectionne ensuite les variables ayant les scores les plus élevés. Ces scores peuvent être basés sur la corrélation, l'information mutuelle, des tests statistiques, etc.
- **Minimal Redundancy Selection** : cette approche vise à éliminer les variables qui sont fortement corrélées entre elles. En effet, deux variables très similaires n'apportent pas d'information complémentaire utile.
- **mRMR (minimum Redundancy Maximum Relevance)** : il s'agit d'une combinaison des deux critères précédents : on cherche à sélectionner les variables à la fois fortement liées à la cible et peu redondantes entre elles.
- **Stepwise Regression / Selection** : méthode gloutonne et itérative qui ajoute ou retire des variables en fonction de leur contribution à la performance du modèle. Le critère de sélection peut être basé sur un test de vraisemblance (likelihood ratio test), avec arrêt lorsque l'ajout ou le retrait d'une variable ne génère plus d'amélioration significative.
- **L1 Regularization (Lasso)** : cette méthode repose sur l'ajout d'un terme de pénalisation proportionnel à la norme L1 des coefficients dans la fonction de perte. Elle pousse certains coefficients à exactement zéro, réalisant ainsi une sélection implicite des variables. L'avantage est qu'elle s'effectue directement pendant l'entraînement du modèle, sans nécessiter de prétraitement.
- **Méthodes embarquées** : certains modèles réalisent automatiquement une sélection de variables, comme les arbres de décision ou les méthodes ensemblistes comme le bagging (e.g., Random Forest), qui attribuent des importances aux variables en fonction de leur utilité dans les décisions prises.

Question 3 - CCA

Describe and derive in detail (mathematically) the method allowing to perform a canonical correlation analysis. Interpret the results, e.g., what is the interpretation of the eigenvalues? How can we obtain the coordinates of the data in the principal components system? Finally, describe some methods for performing feature selection (and not extraction).

Describe and derive in detail (mathematically) the method allowing to perform a canonical correlation analysis.

L'Analyse Canonique (CCA) est une méthode statistique multivariée utilisée pour analyser les relations entre deux ensembles de variables aléatoires, notés ici X et Y , qui représentent des réalisations de vecteurs aléatoires x et y .

Le but de la CCA est de trouver des combinaisons linéaires $z_x = u_x^T x$ et $z_y = u_y^T y$ qui soient **maximamente corrélées**, et qui définissent un espace des scores permettant de capturer et d'expliquer au mieux les relations linéaires entre les deux ensembles de variables. C'est-à-dire :

$$\max_{u_x, u_y} \{ \text{cov}(z_x, z_y) \} \quad \text{sous contrainte que } \sigma_{z_x}^2 = 1 \quad \text{et} \quad \sigma_{z_y}^2 = 1.$$

Nous fixons les variances à 1 afin d'avoir un problème bien posé, d'assurer que chaque variable projetée ait le même poids dans l'optimisation, et de faire en sorte que la covariance devienne une corrélation, ce qui en facilite l'interprétation.

En centrant les variables (i.e., $\tilde{x} = x - \mathbb{E}[x]$, $\tilde{y} = y - \mathbb{E}[y]$), on obtient :

$$\tilde{z}_x = u_x^T \tilde{x}, \quad \tilde{z}_y = u_y^T \tilde{y}.$$

La covariance entre les deux scores est alors :

$$\text{cov}(z_x, z_y) = \mathbb{E}[\tilde{z}_x \tilde{z}_y] = \mathbb{E}[u_x^T \tilde{x} \cdot u_y^T \tilde{y}] = u_x^T \mathbb{E}[\tilde{x} \tilde{y}^T] u_y = u_x^T S_{xy} u_y,$$

où S_{xy} est la matrice de covariance croisée entre x et y .

De même, les variances sont :

$$\sigma_{z_x}^2 = u_x^T S_{xx} u_x, \quad \sigma_{z_y}^2 = u_y^T S_{yy} u_y,$$

avec :

- $S_{xx} = \mathbb{E}[(x - g_x)(x - g_x)^T]$,
- $S_{yy} = \mathbb{E}[(y - g_y)(y - g_y)^T]$,
- $S_{xy} = \mathbb{E}[(x - g_x)(y - g_y)^T]$ (non carrée si x et y ont des dimensions différentes).

On peut estimer les matrices de covariance à partir des données empiriques. Par exemple, pour la covariance croisée entre x et y , on utilise la formule suivante, avec n le nombre d'observations :

$$\Sigma_{xy} = \frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})(y_k - \bar{y})^T$$

où :

- x_k et y_k sont les observations de x et y ,
- \bar{x} et \bar{y} sont les moyennes empiriques des variables x et y .

Les autres matrices de covariance (Σ_{xx} , Σ_{yy}) s'estiment de manière similaire.

Formulation du problème

On cherche à maximiser :

$$\max_{u_x, u_y} \{ u_x^T \Sigma_{xy} u_y \}$$

sous les contraintes :

$$u_x^T \Sigma_{xx} u_x = 1, \quad u_y^T \Sigma_{yy} u_y = 1.$$

Méthode de Lagrange

On définit la fonction de Lagrange :

$$\mathcal{L} = u_x^T \Sigma_{xy} u_y - \frac{\lambda_x}{2} (u_x^T \Sigma_{xx} u_x - 1) - \frac{\lambda_y}{2} (u_y^T \Sigma_{yy} u_y - 1).$$

On dérive par rapport à u_x et u_y :

$$\frac{\partial \mathcal{L}}{\partial u_x} = \Sigma_{xy} u_y - \lambda_x \Sigma_{xx} u_x = 0 \Leftrightarrow \Sigma_{xy} u_y = \lambda_x \Sigma_{xx} u_x,$$

$$\frac{\partial \mathcal{L}}{\partial u_y} = \Sigma_{yx} u_x - \lambda_y \Sigma_{yy} u_y = 0 \Leftrightarrow \Sigma_{yx} u_x = \lambda_y \Sigma_{yy} u_y.$$

En prémultipliant la première équation par u_x^T et la seconde par u_y^T , on montre que :

$$\lambda_x = \lambda_y = \text{cov}(z_x, z_y), \quad \Rightarrow \lambda = \lambda_x \lambda_y = \text{cov}^2(z_x, z_y) \geq 0.$$

Problème aux valeurs propres

On en déduit un problème aux valeurs propres :

$$\Sigma_{xx}^{-1} \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx} u_x = \lambda u_x, \quad \text{ou bien} \quad \Sigma_{yy}^{-1} \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy} u_y = \lambda u_y.$$

Remarque : Comme chaque vecteur peut être obtenu à partir de l'autre via :

$$u_x = \frac{1}{\lambda} \Sigma_{xx}^{-1} \Sigma_{xy} u_y, \quad u_y = \frac{1}{\lambda} \Sigma_{yy}^{-1} \Sigma_{yx} u_x,$$

il suffit de résoudre un seul des deux systèmes spectraux.

Chaque paire canonique (ou composante canonique) représente un nouveau mode de corrélation linéaire maximale entre les deux ensembles de variables X et Y , sous la contrainte d'être orthogonale (décorrélée) aux paires précédentes. Cela garantit que chaque paire apporte une information nouvelle, indépendante des précédentes.

Interpret the results, e.g., what is the interpretation of the eigenvalues?

Les **valeurs propres** λ_i issues de l'analyse canonique représentent le **carré de la corrélation canonique** entre les couples de variables canoniques $z_{x,i} = u_{x,i}^T x$ et $z_{y,i} = u_{y,i}^T y$. Autrement dit :

$$\lambda_i = \text{cov}^2(z_{x,i}, z_{y,i}).$$

Ainsi :

- Une grande valeur propre (proche de 1) indique une forte corrélation entre les combinaisons linéaires de X et Y , i.e., les deux ensembles partagent une structure commune.
- Une petite valeur propre (proche de 0) suggère qu'il existe peu ou pas de relation linéaire entre les combinaisons correspondantes.

Les couples de variables canoniques sont ordonnés selon les valeurs propres décroissantes. Chaque nouveau couple est construit de manière à être orthogonal (décorrélé) aux précédents, garantissant une information non redondante dans chaque direction canonique.

Plus les premières valeurs propres sont élevées, plus les deux ensembles X et Y partagent une dépendance linéaire forte.

How can we obtain the coordinates of the data in the principal components system?

Une fois $u_{x,1}$ déterminé (le vecteur propre associé à la plus grande valeur propre), score de la première composante canonique pour une observation x_k est donné (à un facteur d'échelle près) par :

$$z_{x,1} = u_{x,1}^T (x_k - g_x), \quad \text{et de même} \quad z_{y,1} = u_{y,1}^T (y_k - g_y).$$

Ces scores représentent la projection des observations sur les directions canoniques maximisant la corrélation entre les deux ensembles de variables.

Finally, describe some methods for performing feature selection (and not extraction).

Les méthodes de feature selection, contrairement aux méthodes de feature extraction, ne projettent pas les données dans un espace de dimension réduite et ne transforment donc pas les variables d'origine. Elles consistent plutôt à éliminer les variables les moins pertinentes pour la tâche d'apprentissage.

Ces méthodes sont utilisées pour :

- Lutter contre la malédiction de la dimensionnalité (curse of dimensionality)
- Améliorer la généralisation des modèles en réduisant le risque de surapprentissage (overfitting)
- accélérer le processus d'entraînement
- améliorer l'interprétabilité des modèles

Il existe plusieurs approches principales :

- **Maximum Relevance Selection** : cette méthode calcule un score mesurant la pertinence de chaque variable par rapport à la variable cible. On sélectionne ensuite les variables ayant les scores les plus élevés. Ces scores peuvent être basés sur la corrélation, l'information mutuelle, des tests statistiques, etc.
- **Minimal Redundancy Selection** : cette approche vise à éliminer les variables qui sont fortement corrélées entre elles. En effet, deux variables très similaires n'apportent pas d'information complémentaire utile.
- **mRMR (minimum Redundancy Maximum Relevance)** : il s'agit d'une combinaison des deux critères précédents : on cherche à sélectionner les variables à la fois fortement liées à la cible et peu redondantes entre elles.
- **Stepwise Regression / Selection** : méthode gloutonne et itérative qui ajoute ou retire des variables en fonction de leur contribution à la performance du modèle. Le critère de sélection peut être basé sur un test de vraisemblance (likelihood ratio test), avec arrêt lorsque l'ajout ou le retrait d'une variable ne génère plus d'amélioration significative.
- **L1 Regularization (Lasso)** : cette méthode repose sur l'ajout d'un terme de pénalisation proportionnel à la norme L1 des coefficients dans la fonction de perte. Elle pousse certains coefficients à exactement zéro, réalisant ainsi une sélection implicite des variables. L'avantage est qu'elle s'effectue directement pendant l'entraînement du modèle, sans nécessiter de prétraitement.
- **Méthodes embarquées** : certains modèles réalisent automatiquement une sélection de variables, comme les arbres de décision ou les méthodes ensemblistes comme le bagging (e.g., Random Forest), qui attribuent des importances aux variables en fonction de leur utilité dans les décisions prises.

Question 4 - MCA

Describe and derive in detail (mathematically) the method allowing to perform a multiple correspondence analysis. Interpret the results, e.g., what is the interpretation of the eigenvalues? How can we obtain the coordinates of the data in the principal components system? Finally, describe some methods for performing feature selection (and not extraction).

Describe and derive in detail (mathematically) the method allowing to perform a multiple correspondence analysis.

Cette technique d'extraction de caractéristiques est utilisée lorsque les variables sont catégorielles. C'est l'analogue de PCA pour les variables catégorielles.

On associe à chaque variable x_i un vecteur \mathbf{x}^i qui est un vecteur de codage one-hot pour cette variable. On considère p variables catégorielles. On cherche à trouver une combinaison linéaire des éléments telle que $y_i = u_i^T \mathbf{x}^i$, qui maximise la somme des associations entre les variables catégorielles.

Cela revient à déterminer les axes qui préservent les relations les plus fortes entre les variables.

Remarque 1 : L'Analyse des Correspondances Multiples (MCA) est une généralisation de l'Analyse des Correspondances Simples (SCA), laquelle s'applique uniquement à deux variables qualitatives. L'ACM étend ce cadre au cas où l'on observe simultanément plus de deux variables catégorielles ($p \geq 2$), afin d'en extraire une représentation synthétique et interprétable des relations multivariées entre modalités.

Remarque 2 : dans ce cas, les données ne sont pas centrées mais la matrice de covariance par définition si.

On cherche les combinaisons linéaires qui maximisent la corrélation entre les variables, telle que quantifiée par la somme des variances :

$$\sum_{i=1}^p \sum_{j=1}^p \text{cov}(y_i, y_j) = \sum_{i=1}^p \sum_{j=1}^p \mathbb{E}[y_i y_j] = \mathbb{E} \left[\sum_{i=1}^p \sum_{j=1}^p y_i y_j \right] = \mathbb{E} \left[\left(\sum_{i=1}^p y_i \right)^2 \right]$$

sous la contrainte que la somme des variances est constante, par exemple égale à p :

$$\sum_{i=1}^p \text{cov}(y_i, y_i) = \sum_{i=1}^p \sigma_{y_i}^2 = p.$$

L'objectif est d'imposer que les variables soient sur la même échelle en variance, afin qu'elles contribuent de manière équitable à l'analyse.

Ainsi, on cherche à :

$$\max_{u_1, u_2, \dots} \left\{ \mathbb{E} \left[\left(\sum_{i=1}^p y_i \right)^2 \right] \right\}.$$

On exprime maintenant la somme des covariances sous forme matricielle :

$$\mathbb{E} \left[\left(\sum_{i=1}^p y_i \right)^2 \right] = \mathbb{E} \left[\left(\sum_{i=1}^p u_i^T \mathbf{x}^i \right)^2 \right] = \mathbb{E} [(u^T x)^2] = u^T \mathbb{E}[x x^T] u = u^T F u.$$

La matrice $F = \mathbb{E}[x x^T]$, appelé tableau de contingence normalisé généralisé, contient les éléments f_{kl} qui représentent la probabilité que les éléments k et l apparaissent conjointement (rappelons que les variables sont catégorielles). Cette probabilité peut être estimée empiriquement par :

$$f_{kl} = \frac{n_{kl}}{\sum_k \sum_l n_{kl}} = \frac{n_{kl}}{n},$$

où n_{kl} est le nombre de fois où les deux attributs sont apparus ensemble dans les données.

En d'autre mots, F est la matrice de similarité globale entre les modalités, via les co-occurrences empirique.

On calcule maintenant les variances :

$$\sigma_{y_i}^2 = \mathbb{E}[y_i y_i] = \mathbb{E}[u_i^T x_i \cdot u_i^T x_i] = u_i^T \mathbb{E}[x_i x_i^T] u_i = u_i^T D_i u_i,$$

où la matrice D_i est diagonale, car deux attributs d'une même variable ne peuvent pas apparaître simultanément. Les éléments diagonaux $[D_i]_{kk} = \mathbb{E}[x_i^k]$ représentent les probabilités a priori d'observer l'attribut k , et peuvent être estimés à partir des données comme étant la fréquence de l'attribut. C'est donc la fréquence de la modalité k . La somme des variances devient alors :

$$\sum_{i=1}^p \sigma_{y_i}^2 = \sum_{i=1}^p u_i^T D_i u_i = u^T D u,$$

où D est une grande matrice diagonale contenant les matrices D_i en bloc-diagonal. Cela correspond donc au fréquence marginale des modalités et donc reflète les modalités importantes.

On construit alors la fonction de Lagrange :

$$\mathcal{L} = u^T F u + \lambda(p - u^T D u).$$

En prenant le gradient et en l'annulant, on obtient :

$$F u - \lambda D u = 0 \Rightarrow D^{-1} F u = \lambda u.$$

On retrouve ainsi un problème aux valeurs propres. En multipliant cette équation par u^T , on obtient :

$$\lambda = \frac{u^T F u}{p} > 0.$$

λ est donc proportionnelle à la moyenne des covariances entre les variables.

Encore une fois, il faut sélectionner les vecteurs propres associés aux plus grandes valeurs propres.

Interpret the results, e.g., what is the interpretation of the eigenvalues?

Les valeurs propres sont proportionnelles à la moyenne des covariances entre les variables.

Le premier vecteur propre u_1 est celui associé à la plus grande valeur propre, c'est-à-dire qu'il maximise la somme des covariances entre les variables.

Puisque les x_i sont des variables indicatrices binaires (one-hot), les éléments du vecteur propre dominant u , c'est-à-dire celui associé à la plus grande valeur propre, représentent les **scores attribués à chaque modalité** (ou attribut) des différentes variables sur le **premier axe factoriel**.

Autrement dit :

- chaque composante de u est associée à une modalité spécifique d'une variable catégorielle ;
- ces composantes indiquent dans quelle mesure chaque modalité contribue à l'axe factoriel ;
- ces scores permettent ainsi de **représenter graphiquement les modalités des différentes variables catégorielles** dans un même plan (2D ou 3D), généralement via les premières composantes principales ;
- on obtient ainsi une visualisation conjointe des modalités, ce qui facilite l'analyse des relations entre les catégories.

How can we obtain the coordinates of the data in the principal components system?

Une fois les vecteurs propres $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ extraits (ceux associés aux plus grandes valeurs propres), on peut projeter les individus dans le nouveau système de coordonnées défini par les axes factoriels.

Soit X la matrice des données (de taille $n \times m$, où chaque ligne représente un individu et chaque colonne une modalité codée en one-hot), alors la projection des individus dans le nouvel espace s'écrit :

$$Z = X u,$$

où u est la matrice contenant les vecteurs propres sélectionnés en colonnes (de taille $m \times k$) et Z est la matrice projetée (de taille $n \times k$), contenant les coordonnées des individus dans le système des composantes principales.

Ces coordonnées sont appelées **scores des individus** sur les axes factoriels. Elles permettent de visualiser les relations entre individus dans un espace de plus faible dimension tout en préservant au mieux les associations entre modalités.

Cette projection est utilisée pour :

- représenter les individus dans un plan factoriel (généralement les deux premiers axes),
- interpréter les regroupements ou proximités entre individus,
- détecter les axes les plus explicatifs de la variabilité des réponses.

Finally, describe some methods for performing feature selection (and not extraction).

Les méthodes de feature selection, contrairement aux méthodes de feature extraction, ne projettent pas les données dans un espace de dimension réduite et ne transforment donc pas les variables d'origine. Elles consistent plutôt à éliminer les variables les moins pertinentes pour la tâche d'apprentissage.

Ces méthodes sont utilisées pour :

- Lutter contre la malédiction de la dimensionnalité (curse of dimensionality)
- Améliorer la généralisation des modèles en réduisant le risque de surapprentissage (overfitting)
- accélérer le processus d'entraînement
- améliorer l'interprétabilité des modèles

Il existe plusieurs approches principales :

- **Maximum Relevance Selection** : cette méthode calcule un score mesurant la pertinence de chaque variable par rapport à la variable cible. On sélectionne ensuite les variables ayant les scores les plus élevés. Ces scores peuvent être basés sur la corrélation, l'information mutuelle, des tests statistiques, etc.
- **Minimal Redundancy Selection** : cette approche vise à éliminer les variables qui sont fortement corrélées entre elles. En effet, deux variables très similaires n'apportent pas d'information complémentaire utile.
- **mRMR (minimum Redundancy Maximum Relevance)** : il s'agit d'une combinaison des deux critères précédents : on cherche à sélectionner les variables à la fois fortement liées à la cible et peu redondantes entre elles.
- **Stepwise Regression / Selection** : méthode gloutonne et itérative qui ajoute ou retire des variables en fonction de leur contribution à la performance du modèle. Le critère de sélection peut être basé sur un test de vraisemblance (likelihood ratio test), avec arrêt lorsque l'ajout ou le retrait d'une variable ne génère plus d'amélioration significative.
- **L1 Regularization (Lasso)** : cette méthode repose sur l'ajout d'un terme de pénalisation proportionnel à la norme L1 des coefficients dans la fonction de perte. Elle pousse certains coefficients à exactement zéro, réalisant ainsi une sélection implicite des variables. L'avantage est qu'elle s'effectue directement pendant l'entraînement du modèle, sans nécessiter de prétraitement.
- **Méthodes embarquées** : certains modèles réalisent automatiquement une sélection de variables, comme les arbres de décision ou les méthodes ensemblistes comme le bagging (e.g., Random Forest), qui attribuent des importances aux variables en fonction de leur utilité dans les décisions prises.

Question 5 - MDS

Derive (mathematically) and explain the technique of classical multidimensional scaling. Prove the formula allowing to compute the inner products from the distances and then the distances from the inner products. Then, describe how a data matrix can be obtained from an inner product matrix. Describe informally the links with principal components analysis. What is finally the procedure for drawing the data from their distances?

Derive (mathematically) and explain the technique of classical multidimensional scaling.

La méthode MDS (Multidimensional Scaling) est utilisée pour représenter n échantillons entre lesquels on a une notion de distance ou de proximité, dans un espace euclidien. Elle repose sur les distances entre paires d'échantillons.

On suppose que l'on dispose d'une matrice D ou S , représentant respectivement les distances euclidienne ou les similarités (produit scalaire, similarité cosinus, corrélation). On définit également une matrice K , contenant les produits scalaires entre les vecteurs de caractéristiques x_i (souvent appelée matrice de Gram) :

$$k_{ij} = K_{ij} = x_i^T x_j \Rightarrow K = XX^T$$

où X est la matrice contenant les vecteurs de caractéristiques x_i en lignes.

On peut également convertir des similarités en distances avec :

$$d_{ij} = (s_{ii} - 2s_{ij} + s_{jj})^{1/2}$$

La méthode MDS fonctionne en deux étapes :

1. À partir de D , retrouver la matrice des produits scalaires K .
2. À partir de K , retrouver X de manière à représenter les données tout en préservant les produits scalaires, et donc les distances.

Note: $D \Rightarrow K \Rightarrow X$ is multidimensional scaling.

Prove the formula allowing to compute the inner products from the distances and then the distances from the inner products.

Distances à partir des produits scalaires ($K \Rightarrow D$)

Si les vecteurs de caractéristiques x_i sont connus, on peut calculer :

$$k_{ij} = x_i^T x_j \quad \text{et} \quad d_{ij}^2 = \|x_i - x_j\|^2$$

Si les distances sont euclidiennes, il existe un espace d'embedding dans lequel :

$$d_{ij}^2 = \|x_i - x_j\|^2 = \|x_i\|^2 + \|x_j\|^2 - 2x_i^T x_j = k_{ii} + k_{jj} - 2k_{ij}$$

Sous forme matricielle, on a :

$$D = \text{diag}(K) \cdot \mathbf{e}^T + \mathbf{e} \cdot \text{diag}(K)^T - 2K$$

où $\text{diag}(K)$ est le vecteur colonne contenant les éléments diagonaux de K , et \mathbf{e} est un vecteur colonne de n éléments tous égaux à 1.

Produits scalaires à partir des distances ($D \Rightarrow K$)

On souhaite maintenant retrouver K à partir de D . Ce problème admet plusieurs solutions, car le produit scalaire dépend de l'origine du système de coordonnées.

On suppose donc que K est centrée, c'est-à-dire :

$$K\mathbf{e} = 0 \quad \text{et} \quad \mathbf{e}^T K = 0$$

Ce qui revient à positionner l'origine au centroïde des données.

On calcule alors K selon la formule :

$$K = -\frac{1}{2}H\mathbf{D}\mathbf{H}$$

où $H = I - \frac{1}{n}\mathbf{e}\mathbf{e}^T$ est la matrice de centrage. Lorsque H est appliquée à un vecteur, elle le centre. Pour une matrice M , on a alors $\mathbf{e}^T H M H \mathbf{e} = 0$ et $H M H \mathbf{e} = 0$ grâce à $H\mathbf{e} = 0$, sachant que $\mathbf{e}^T \mathbf{e} = n$.

Preuve de la formule précédente

On part de l'expression de D :

$$-\frac{1}{2}H D H = -\frac{1}{2}H (\text{diag}(K)\mathbf{e}^T + \mathbf{e} \text{diag}(K)^T - 2K) H$$

En utilisant $H\mathbf{e} = 0$, on obtient :

$$-\frac{1}{2}H D H = -\frac{1}{2}(0 + 0 - 2H K H) = K \quad (\text{puisque } K \text{ est centrée})$$

Then, describe how a data matrix can be obtained from an inner product matrix.

On suppose que la matrice K des produits scalaires est connue et valide, c'est-à-dire symétrique et semi-définie positive. Par définition, on a :

$$K = X X^T$$

où X est la matrice contenant les vecteurs de caractéristiques à reconstruire.

D'après la théorie spectrale des matrices symétriques, toute matrice K admet une décomposition spectrale :

$$K = U \Lambda U^T$$

où :

- U est une matrice orthogonale dont les colonnes sont les vecteurs propres de K ,
- Λ est une matrice diagonale contenant les valeurs propres associées.

À partir de cette décomposition, on peut réécrire :

$$K = (U \Lambda^{1/2})(U \Lambda^{1/2})^T$$

Ce qui permet de reconstruire la matrice des données :

$$X = U \Lambda^{1/2}$$

Autrement dit, chaque ligne de X représente un point de l'espace euclidien reconstruit à partir de K , en préservant les produits scalaires (et donc les distances si l'espace est euclidien).

Remarque importante : pour que cette construction soit possible, il est nécessaire que toutes les valeurs propres de K soient positives ou nulles. Si certaines valeurs propres sont négatives (ce qui peut arriver à cause du bruit ou d'erreurs dans la matrice de distance initiale), on peut les tronquer à zéro afin d'obtenir une approximation valide dans un espace euclidien.

Describe informally the links with principal components analysis.

Les vecteurs propres de K correspondent aux coordonnées des individus dans le système d'axes principaux de PCA, à un facteur d'échelle près.

Supposons u_1 , le vecteur propre dominant de $X^T X$, obtenu par PCA. Il vérifie :

$$X^T X u_1 = \lambda_1 u_1$$

En multipliant par X à gauche, on obtient :

$$X X^T (X u_1) = \lambda_1 (X u_1)$$

Donc $X u_1$ est un vecteur propre de $K = X X^T$. Or, $X u_1$ est la projection des vecteurs x_i sur le premier axe principal de PCA.

Cela montre que le vecteur propre dominant de K donne les coordonnées des points dans le plan factoriel de PCA. Ce résultat se généralise au i -ème vecteur propre.

Ainsi, on a bien $X = U \Lambda^{1/2}$, exprimé dans la base des composantes principales de PCA.

En d'autres termes, si les distances D sont euclidiennes, MDS effectue exactement le même travail que PCA, mais à partir des distances entre les données, et non à partir des données elles-mêmes.

Cas des valeurs propres négatives

Si K n'est pas semi-définie positive, alors les échantillons ne peuvent pas être représentés dans un espace euclidien tout en préservant les distances. Une approximation courante consiste à remplacer les valeurs propres négatives par 0.

What is finally the procedure for drawing the data from their distances?

- **Étape 1 :** Calculer la matrice des produits scalaires K à partir de la matrice des distances D , en appliquant la formule de double centrage :

$$K = -\frac{1}{2}H D H$$

où $H = I - \frac{1}{n}\mathbf{e}\mathbf{e}^T$ est la matrice de centrage, et \mathbf{e} est le vecteur colonne de taille n rempli de 1.

- **Étape 2 :** Vérifier que K est semi-définie positive, c'est-à-dire que toutes ses valeurs propres sont positives ou nulles. Si ce n'est pas le cas, tronquer les valeurs propres négatives à zéro afin d'obtenir une approximation valide.

- **Étape 3 :** Calculer la matrice des données X , à partir de la décomposition spectrale de K , en utilisant :

$$X = U \Lambda^{1/2}$$

où U est la matrice des vecteurs propres de K , et Λ est la matrice diagonale contenant les valeurs propres (modifiées si nécessaire à l'étape précédente).

Question 6 - Edit Distance (DP)

Derive (mathematically) and explain the dynamic programming recurrence formula and apply it to the problem of comparing two sequences of symbols (edit-distance). As an exercise, solve a concrete example comparing two short sequences (provided at the exam).

Derive (mathematically) and explain the dynamic programming recurrence formula.

La **programmation dynamique (DP)** est une méthode algorithmique utilisée pour résoudre des problèmes d'optimisation en décomposant un problème complexe en sous-problèmes plus simples, dont les résultats sont stockés pour éviter des calculs redondants.

Plutôt que d'explorer toutes les combinaisons possibles (souvent de complexité exponentielle), DP exploite la structure récursive du problème et utilise une **table de mémoire** pour enregistrer les résultats intermédiaires.

Par exemple, dans une **lattice** (graphe en couche) composée de N niveaux avec N états possibles à chaque niveau, une approche naïve impliquerait $N \times N$ évaluations à chaque étape. En utilisant DP, on peut réduire ce coût à seulement N évaluations par niveau, soit un total de $O(N^2)$ au lieu de $O(N^N)$, en évitant de recalculer plusieurs fois le même sous-problème. L'idée globale du problème est que minimiser localement revient à minimiser globalement.

Formulation du problème

On définit d'abord s_k une variable aléatoire, correspondant à l'état à l'étape k , et $d(s_k = j \mid s_{k-1} = i) = d_{ij}$ comme le coût pour passer de l'état i (au temps $k-1$) à l'état j (au temps k).

Le coût total d'un chemin est donné par :

$$D(s_0, s_1, \dots, s_N) = \sum_{i=1}^N d(s_i \mid s_{i-1})$$

Le coût optimal à partir d'un état initial s_0 est :

$$D^*(s_0) = \min_{s_1, \dots, s_N} \left\{ \sum_{i=1}^N d(s_i \mid s_{i-1}) \right\}$$

Le coût optimal global (quel que soit l'état de départ) est :

$$D^* = \min_{s_0} \{D^*(s_0)\}$$

Formules de récurrence de Bellman (Programmation dynamique backward)

$$\begin{cases} D^*(s_N) = 0 \\ D^*(s_k = i) = \min_{s_{k+1}} \{d(s_{k+1} \mid s_k = i) + D^*(s_{k+1})\} \\ D^* = \min_{s_0} \{D^*(s_0)\} \end{cases}$$

Preuve de la formule de récurrence de Bellman :

$$\begin{aligned} D^*(s_k) &= \min_{s_{k+1}, \dots, s_N} \left\{ \sum_{i=k+1}^N d(s_i \mid s_{i-1}) \right\} \\ &= \min_{s_{k+1}, \dots, s_N} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \\ &= \min_{s_{k+1}} \left\{ \min_{s_{k+2}, \dots, s_N} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \right\} \\ &= \min_{s_{k+1}} \left\{ d(s_{k+1} \mid s_k) + \min_{s_{k+2}, \dots, s_N} \left\{ \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \right\} \\ &= \min_{s_{k+1}} \{d(s_{k+1} \mid s_k) + D^*(s_{k+1})\} \end{aligned}$$

Version symétrique : programmation dynamique forward

De manière symétrique, on peut aussi écrire :

$$\begin{cases} D^*(s_0) = 0 \\ D^*(s_k = i) = \min_{s_{k-1}} \{d(s_k = i \mid s_{k-1}) + D^*(s_{k-1})\} \\ D^* = \min_{s_N} \{D^*(s_N)\} \end{cases}$$

Retracer le chemin optimal

Pour non seulement calculer le coût minimal mais également **retrouver le chemin optimal**, il est indispensable de mémoriser, à chaque étape, l'état précédent ayant permis d'atteindre un coût minimal. Cela se fait en conservant, pour chaque état s_k , un **pointeur vers l'état s_{k+1}** qui a permis d'obtenir le coût optimal $D^*(s_k)$. Une fois le coût final $D^*(s_0)$ (ou $D^*(s_N)$ si on travaille en forward) obtenu, il suffit alors de faire un **backtracking** à partir de cet état pour reconstituer le chemin optimal complet.

Adaptation aux graphes orientés acycliques (DAG)

Dans des situations plus générales où l'on travaille sur un **graphhe orienté acyclique (DAG)** sans structure strictement séquentielle (comme dans une lattice régulière), on peut tout de même appliquer la programmation dynamique avec quelques ajustements :

- On peut transformer le DAG en une **lattice $N \times N$** en ajoutant des **nœuds fictifs (dummy nodes)** avec un coût nul, de manière à "aligner" tous les chemins et rendre la structure régulière.
- Alternativement, on peut adapter la formule de récurrence en remplaçant les dépendances strictes entre niveaux k et $k + 1$ par un **ensemble de successeurs** accessibles à partir d'un état donné.

Apply it to the problem of comparing two sequences of symbols (edit-distance).

La **distance d'édition** calcule le nombre minimal de transformations nécessaires pour transformer une chaîne x en une chaîne y , où x et y peuvent avoir des longueurs différentes.

Notations :

- $x_i^{|x|}$ désigne les $|x| - i$ derniers caractères de x , c'est-à-dire $x_i, \dots, x_{|x|-1}$.
- y_0^j désigne les j premiers caractères de y , c'est-à-dire y_0, \dots, y_{j-1} .
- Chaque état est représenté par un couple (i, j) , correspondant à la sous-séquence $x_i^{|x|}$ et y_0^j .
- Le niveau de traitement est défini par $k = i + j$, constant à chaque étape.

On lit la chaîne x de gauche à droite, afin de construire progressivement la chaîne y .

Trois opérations sont possibles :

- **Insertion (1 niveau)** d'un caractère dans y (sans consommer de caractère de x) : coût = 1.
- **Suppression (1 niveau)** d'un caractère de x (sans ajouter à y) : coût = 1.
- **Substitution (2 niveau)** d'un caractère de x par un caractère de y : coût = δ_{ij} (0 si $x_i = y_j$, 1 sinon).

Remarque sur l'adaptation des coûts :

Les coûts associés aux opérations d'insertion, suppression et substitution peuvent être adaptés en fonction du domaine d'application. Par exemple, en **reconnaissance vocale**, la substitution de sons phonétiquement proches (comme /p/ et /b/) peut être pénalisée faiblement, tandis que la substitution de sons très différents (comme /s/ et /u/) peut avoir un coût élevé.

Initialisation :

$$D^*(x_0^{|x|}, y_0^0) = 0$$

Formule de récurrence :

$$D^*(x_i^{|x|}, y_0^j) = \min \begin{cases} D^*(x_i^{|x|}, y_0^{j-1}) + 1 & \text{(insertion)} \\ D^*(x_{i-1}^{|x|}, y_0^j) + 1 & \text{(suppression)} \\ D^*(x_{i-1}^{|x|}, y_0^{j-1}) + \delta_{ij} & \text{(substitution)} \end{cases}$$

Résultat final :

$$\text{edit_dist}(x, y) = D^*(x_{|x|}^{|x|}, y_0^{|y|})$$

Implémentation pratique.

Lors de l'implémentation de l'algorithme, on remplit la table **niveau par niveau**, c'est-à-dire en suivant les **diagonales de la matrice** (où le niveau est défini comme $k = i + j$, pour les indices de ligne et de colonne). Cela permet de respecter les dépendances de la récurrence.

Lien avec la plus longue sous-séquence commune (LCS)

Une quantité intimement liée à la distance d'édition est la **longest common subsequence** (LCS), notée $\text{lcs}(x, y)$. Il existe une relation simple entre la distance d'édition et la LCS, dans le cas particulier où le coût d'une substitution est deux fois plus élevé que le coût d'une insertion et d'une suppression.

Dans ce cadre, on peut montrer que :

$$\text{edit_dist}(x, y) = |x| + |y| - 2 \cdot \text{lcs}(x, y) \Rightarrow \text{lcs}(x, y) = \frac{1}{2} (|x| + |y| - \text{edit_dist}(x, y))$$

As an exercise, solve a concrete example comparing two short sequences (provided at the exam).

On veut transformer la chaîne $x = \text{livre}$ en $y = \text{lire}$.

Étapes de l'algorithme :

On utilise une matrice D de taille $(|x| + 1) \times (|y| + 1)$, ici 6×5 , car on ajoute une ligne et une colonne pour représenter les chaînes vides.

Règles de coût :

- Insertion : 1
- Suppression : 1
- Substitution : $\delta_{ij} = 0$ si $x_i = y_j$, sinon 1

Résolution :

On initialise la première ligne avec les indices de colonnes (insertions successives) et la première colonne avec les indices de lignes (suppressions successives) et puis on applique la récurrence à chaque cellule $D_{i,j}$, selon les 3 cas : insertion, suppression, substitution.

	\emptyset	l	i	r	e	
\emptyset	0	1	2	3	4	
l	1					
i	2					
v	3					
r	4					
e	5					

	\emptyset	l	i	r	e	
\emptyset	0	1	2	3	4	
l	1	0	1	2	3	
i	2	1	0	1	2	
v	3	2	1	1	2	
r	4	3	2	1	2	
e	5	4	3	2	1	

Résultat :

La distance d'édition entre **livre** et **lire** est donc :

$$\text{edit_dist}(\text{livre}, \text{lire}) = D_{5,4} = \boxed{1}$$

Chemin optimal (backtracking) :

On suit les opérations qui mènent au coût minimum :

- **livre** → **lire** (on supprime le v au milieu)

Une seule suppression permet d'aligner parfaitement les deux chaînes.

Question 7 - Bellman-Ford (DP)

Derive (mathematically) and explain the dynamic programming recurrence formula and apply it to the problem of computing the shortest-path distance between two nodes of a weighted directed graph (Bellman-Ford algorithm). As an exercise, solve a concrete example on a small graph (provided at the exam).

Derive (mathematically) and explain the dynamic programming recurrence formula.

La **programmation dynamique (DP)** est une méthode algorithmique utilisée pour résoudre des problèmes d'optimisation en décomposant un problème complexe en sous-problèmes plus simples, dont les résultats sont stockés pour éviter des calculs redondants.

Plutôt que d'explorer toutes les combinaisons possibles (souvent de complexité exponentielle), DP exploite la structure récursive du problème et utilise une **table de mémoire** pour enregistrer les résultats intermédiaires.

Par exemple, dans une **lattice** (graphe en couche) composée de N niveaux avec N états possibles à chaque niveau, une approche naïve impliquerait $N \times N$ évaluations à chaque étape. En utilisant DP, on peut réduire ce coût à seulement N évaluations par niveau, soit un total de $O(N^2)$ au lieu de $O(N^N)$, en évitant de recalculer plusieurs fois le même sous-problème. L'idée globale du problème est que minimiser localement revient à minimiser globalement.

Formulation du problème

On définit d'abord s_k une variable aléatoire, correspondant à l'état à l'étape k , et $d(s_k = j \mid s_{k-1} = i) = d_{ij}$ comme le coût pour passer de l'état i (au temps $k-1$) à l'état j (au temps k).

Le coût total d'un chemin est donné par :

$$D(s_0, s_1, \dots, s_N) = \sum_{i=1}^N d(s_i \mid s_{i-1})$$

Le coût optimal à partir d'un état initial s_0 est :

$$D^*(s_0) = \min_{s_1, \dots, s_N} \left\{ \sum_{i=1}^N d(s_i \mid s_{i-1}) \right\}$$

Le coût optimal global (quel que soit l'état de départ) est :

$$D^* = \min_{s_0} \{D^*(s_0)\}$$

Formules de récurrence de Bellman (Programmation dynamique backward)

$$\begin{cases} D^*(s_N) = 0 \\ D^*(s_k = i) = \min_{s_{k+1}} \{d(s_{k+1} \mid s_k = i) + D^*(s_{k+1})\} \\ D^* = \min_{s_0} \{D^*(s_0)\} \end{cases}$$

Preuve de la formule de récurrence de Bellman :

$$\begin{aligned} D^*(s_k) &= \min_{s_{k+1}, \dots, s_N} \left\{ \sum_{i=k+1}^N d(s_i \mid s_{i-1}) \right\} \\ &= \min_{s_{k+1}, \dots, s_N} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \\ &= \min_{s_{k+1}} \left\{ \min_{s_{k+2}, \dots, s_N} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \right\} \\ &= \min_{s_{k+1}} \left\{ d(s_{k+1} \mid s_k) + \min_{s_{k+2}, \dots, s_N} \left\{ \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \right\} \\ &= \min_{s_{k+1}} \{d(s_{k+1} \mid s_k) + D^*(s_{k+1})\} \end{aligned}$$

Version symétrique : programmation dynamique forward

De manière symétrique, on peut aussi écrire:

$$\begin{cases} D^*(s_0) = 0 \\ D^*(s_k = i) = \min_{s_{k-1}} \{d(s_k = i \mid s_{k-1}) + D^*(s_{k-1})\} \\ D^* = \min_{s_N} \{D^*(s_N)\} \end{cases}$$

Retracer le chemin optimal

Pour non seulement calculer le coût minimal mais également **retrouver le chemin optimal**, il est indispensable de mémoriser, à chaque étape, l'état précédent ayant permis d'atteindre un coût minimal. Cela se fait en conservant, pour chaque état s_k , un **pointeur vers l'état s_{k+1}** qui a permis d'obtenir le coût optimal $D^*(s_k)$. Une fois le coût final $D^*(s_0)$ (ou $D^*(s_N)$ si on travaille en forward) obtenu, il suffit alors de faire un **backtracking** à partir de cet état pour reconstituer le chemin optimal complet.

Adaptation aux graphes orientés acycliques (DAG)

Dans des situations plus générales où l'on travaille sur un **graphe orienté acyclique (DAG)** sans structure strictement séquentielle (comme dans une lattice régulière), on peut tout de même appliquer la programmation dynamique avec quelques ajustements :

- On peut transformer le DAG en une **lattice $N \times N$** en ajoutant des **nœuds fictifs (dummy nodes)** avec un coût nul, de manière à "aligner" tous les chemins et rendre la structure régulière.
- Alternativement, on peut adapter la formule de récurrence en remplaçant les dépendances strictes entre niveaux k et $k + 1$ par un **ensemble de successeurs** accessibles à partir d'un état donné.

Apply it to the problem of computing the shortest-path distance between two nodes of a weighted directed graph (Bellman-Ford algorithm).

Supposons que nous disposons d'un graphe orienté pondéré, totalement connecté, c'est-à-dire que chaque noeud est accessible à partir de tous les autres.

L'objectif est d'atteindre le noeud destination 0 depuis n'importe quel noeud avec le chemin de coût minimal.

On utilise la notation $c_{ij} = d(j \mid i)$ pour désigner le coût de transition de i vers j . Les règles suivantes s'appliquent :

- $c_{ij} > 0$ si un lien existe entre i et j , avec $i \neq 0$
- $c_{ij} = \infty$ s'il n'existe pas de lien entre i et j
- $c_{0j} = \infty \quad \forall j \neq 0$ et $c_{00} = 0$: le noeud destination est absorbant (aucune sortie)
- $c_{ii} = \infty \quad \forall i \neq 0$: les boucles (self-loop) ne sont pas autorisées

Déroulement temporel et niveaux :

On considère des niveaux k qui correspondent à des étapes temporelles. À un niveau donné, on se trouve dans un seul noeud. Si n est le nombre total de noeuds, alors il ne peut pas y avoir plus de $n - 1$ niveaux, sinon cela implique nécessairement la revisite d'au moins un noeud, ce qui le rend non optimal dans le cadre d'une recherche de plus court chemin. Ce déroulement permet de créer un DAG (Directed Acyclic Graph) et de pouvoir utiliser l'algorithme Backward DP.

On définit la table $D^*(i, k)$ de dimensions $n \times n$, où i est l'indice du noeud au niveau k . Cette table contient le coût minimal pour atteindre le noeud destination 0 à partir du noeud i , en partant de l'étape k ($D^*(s_k = i)$).

Initialisation :

$$\begin{cases} D^*(0, n - 1) = 0 \\ D^*(i, n - 1) = \infty \quad \forall i \neq 0 \end{cases}$$

Formule de récurrence de Bellman (Programmation dynamique backward) :

$$D^*(i, k) = \min_{j \in \text{Succ}(i)} \{c_{ij} + D^*(j, k + 1)\}$$

Interprétation pratique :

En pratique, on commence par la fin du graphe « déroulé » (déplié dans le temps) et on calcule à rebours le coût pour atteindre la destination (le noeud 0). Cela permet d'évaluer de manière dynamique, pour chaque niveau temporel k , le coût optimal à partir de chaque noeud.

As an exercise, solve a concrete example on a small graph (provided at the exam).

- Initialisation au niveau $n - 1$
- Itérer $n - 1$ fois sur les $n - 1$ niveaux (k de $n - 2$ à 0)
- Appliquer la formule de récurrence pour tout i sur le niveau k

Voici un exemple de grille résolue :

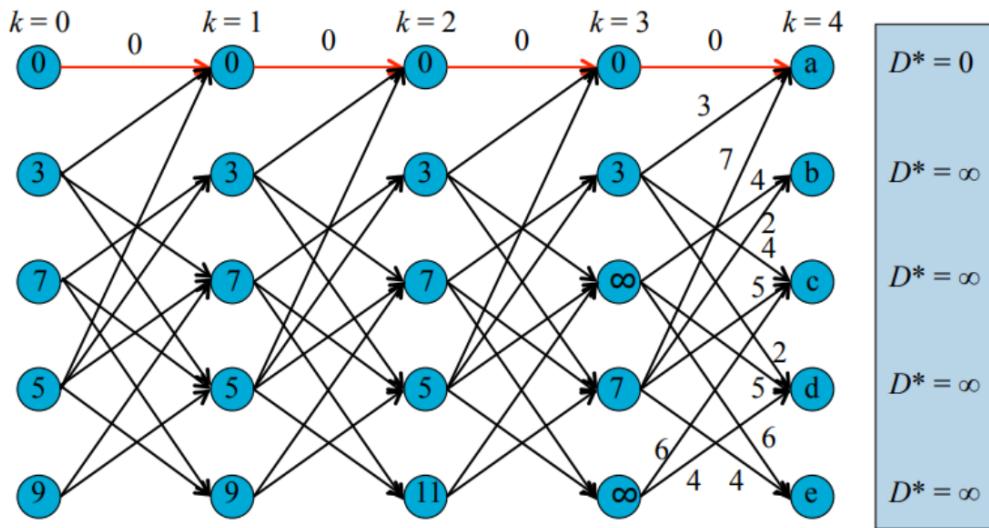


Figure 7.1: Illustration du graphe déroulé pour Bellman-Ford avec programmation dynamique.

Question 8 - DTW

Derive (mathematically) and explain the dynamic programming recurrence formula and explain its application to the problem of « dynamic time warping » in speech recognition.

Derive (mathematically) and explain the dynamic programming recurrence formula.

La **programmation dynamique (DP)** est une méthode algorithmique utilisée pour résoudre des problèmes d'optimisation en décomposant un problème complexe en sous-problèmes plus simples, dont les résultats sont stockés pour éviter des calculs redondants.

Plutôt que d'explorer toutes les combinaisons possibles (souvent de complexité exponentielle), DP exploite la structure récursive du problème et utilise une **table de mémoire** pour enregistrer les résultats intermédiaires.

Par exemple, dans une **lattice** (graphe en couche) composée de N niveaux avec N états possibles à chaque niveau, une approche naïve impliquerait $N \times N$ évaluations à chaque étape. En utilisant DP, on peut réduire ce coût à seulement N évaluations par niveau, soit un total de $O(N^2)$ au lieu de $O(N^N)$, en évitant de recalculer plusieurs fois le même sous-problème. L'idée globale du problème est que minimiser localement revient à minimiser globalement.

Formulation du problème

On définit d'abord s_k une variable aléatoire, correspondant à l'état à l'étape k , et $d(s_k = j \mid s_{k-1} = i) = d_{ij}$ comme le coût pour passer de l'état i (au temps $k-1$) à l'état j (au temps k).

Le coût total d'un chemin est donné par :

$$D(s_0, s_1, \dots, s_N) = \sum_{i=1}^N d(s_i \mid s_{i-1})$$

Le coût optimal à partir d'un état initial s_0 est :

$$D^*(s_0) = \min_{s_1, \dots, s_N} \left\{ \sum_{i=1}^N d(s_i \mid s_{i-1}) \right\}$$

Le coût optimal global (quel que soit l'état de départ) est :

$$D^* = \min_{s_0} \{D^*(s_0)\}$$

Formules de récurrence de Bellman (Programmation dynamique backward)

$$\begin{cases} D^*(s_N) = 0 \\ D^*(s_k = i) = \min_{s_{k+1}} \{d(s_{k+1} \mid s_k = i) + D^*(s_{k+1})\} \\ D^* = \min_{s_0} \{D^*(s_0)\} \end{cases}$$

Preuve de la formule de récurrence de Bellman :

$$\begin{aligned} D^*(s_k) &= \min_{s_{k+1}, \dots, s_N} \left\{ \sum_{i=k+1}^N d(s_i \mid s_{i-1}) \right\} \\ &= \min_{s_{k+1}, \dots, s_N} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \\ &= \min_{s_{k+1}} \left\{ \min_{s_{k+2}, \dots, s_N} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \right\} \\ &= \min_{s_{k+1}} \left\{ d(s_{k+1} \mid s_k) + \min_{s_{k+2}, \dots, s_N} \left\{ \sum_{i=k+2}^N d(s_i \mid s_{i-1}) \right\} \right\} \\ &= \min_{s_{k+1}} \{d(s_{k+1} \mid s_k) + D^*(s_{k+1})\} \end{aligned}$$

Version symétrique : programmation dynamique forward

De manière symétrique, on peut aussi écrire :

$$\begin{cases} D^*(s_0) = 0 \\ D^*(s_k = i) = \min_{s_{k-1}} \{d(s_k = i \mid s_{k-1}) + D^*(s_{k-1})\} \\ D^* = \min_{s_N} \{D^*(s_N)\} \end{cases}$$

Retracer le chemin optimal

Pour non seulement calculer le coût minimal mais également **retrouver le chemin optimal**, il est indispensable de mémoriser, à chaque étape, l'état précédent ayant permis d'atteindre un coût minimal. Cela se fait en conservant, pour chaque état s_k , un **pointeur vers l'état s_{k+1}** qui a permis d'obtenir le coût optimal $D^*(s_k)$. Une fois le coût final $D^*(s_0)$ (ou $D^*(s_N)$ si on travaille en forward) obtenu, il suffit alors de faire un **backtracking** à partir de cet état pour reconstituer le chemin optimal complet.

Adaptation aux graphes orientés acycliques (DAG)

Dans des situations plus générales où l'on travaille sur un **graphe orienté acyclique (DAG)** sans structure strictement séquentielle (comme dans une lattice régulière), on peut tout de même appliquer la programmation dynamique avec quelques ajustements :

- On peut transformer le DAG en une **lattice $N \times N$** en ajoutant des **nœuds fictifs (dummy nodes)** avec un coût nul, de manière à "aligner" tous les chemins et rendre la structure régulière.
- Alternativement, on peut adapter la formule de récurrence en remplaçant les dépendances strictes entre niveaux k et $k + 1$ par un **ensemble de successeurs** accessibles à partir d'un état donné.

Explain its application to the problem of « dynamic time warping » in speech recognition.

La **DTW (Dynamic Time Warping)** est couramment utilisée pour la reconnaissance de mots, par exemple à partir de spectrogrammes. Cependant, il est probable que les mots soient prononcés de manière très différente (en durée, en intonation, etc.). Il est donc nécessaire de tenir compte des distorsions ou *warping* du signal.

L'objectif de la DTW est d'aligner deux signaux temporels (un signal de référence et un signal observé) en minimisant une distance globale, tout en permettant des déformations non linéaires de l'axe temporel.

Alignement temporel et contraintes

Les signaux sont alignés à l'aide d'une distance $d(i, j)$ (Euclidienne par exemple) mesurée entre deux trames (ou fenêtres temporelles) et d'un alignement temporel qui permet un *warping*. Pour garantir que cet alignement forme un DAG et fournit des correspondances pertinentes, certaines contraintes doivent être respectées :

- **Monotonie** : $i_k \geq i_{k-1}, \quad j_k \geq j_{k-1}$

Cette contrainte garantit que l'on ne revient jamais en arrière dans le temps.

- **Continuité** : $i_k - i_{k-1} \leq 1, \quad j_k - j_{k-1} \leq 1$

Cela signifie qu'on peut avancer d'un pas au plus à chaque itération, soit horizontalement, verticalement, ou en diagonale.

- **Conditions aux bords** : $i_1 = j_1 = 1$ et $i_K = I, \quad j_K = J$

L'alignement commence au début des deux signaux et se termine à la fin.

Formule de récurrence

Le problème peut-être résolu par DP en tenant compte uniquement des transitions valides.

Soit $R_n = (r_1^n, \dots, r_I^n)$ le signal de référence et $O = (o_1, \dots, o_J)$ le signal observé. La matrice de coût cumulé $g(i, j)$ est définie récursivement par :

$$\begin{cases} g(1, 1) &= d(r_1^n, o_1) \\ g(i, j) &= \min \begin{cases} g(i - 1, j) + d(r_i^n, o_j) & \text{(insertion)} \\ g(i - 1, j - 1) + 2d(r_i^n, o_j) & \text{(diagonale)} \\ g(i, j - 1) + d(r_i^n, o_j) & \text{(suppression)} \end{cases} \end{cases}$$

où $g(i, j)$ est le coût d'alignement entre $r_1^n \dots r_i^n$ et $o_1 \dots o_j$.

Le coût total de l'alignement est donné par :

$$D(R_n, O) = \frac{1}{I + J} \cdot g(I, J)$$

Remarque : Le facteur 2 dans le cas de l'alignement diagonal est purement empirique : il donne de meilleurs résultats en pratique. La structure de la récurrence est similaire à celle utilisée pour la distance d'édition.

Question 9 et 10 - MDP / QLearning

Explain the general principles behind « Markov decision processes » and derive in detail (mathematically) the expressions allowing to compute the optimal policy (value-iteration technique). Briefly discuss the links with dynamic programming, the Q-value, reinforcement learning and Q-learning (without the proofs this time).

Explain the general principles behind « Markov decision processes » and derive in detail (mathematically) the expressions allowing to compute the optimal policy (value-iteration technique).

Nous considérons un ensemble d'états $S = \{1, 2, \dots, n\}$ tel que $s_t = k$ signifie que le processus est dans l'état k au temps t . Chaque état dispose d'un ensemble d'actions admissibles $U(k)$, telles que $a \in U(k)$ est une action possible en état k .

Cette action dépend uniquement de l'état courant (hypothèse de Markov), c'est-à-dire qu'elle est indépendante du temps et des états précédents.

Lorsque l'on choisit une action $a = u(s_t)$, cela engendre un coût borné $0 \leq c(u(s_t) | s_t) < \infty$, et le système passe à un nouvel état $s_{t+1} = k'$ avec une probabilité donnée par :

$$P(s_{t+1} = k' | s_t = k, u(s_t) = a) = p(k' | k, a)$$

Cette probabilité ne dépend que de l'état courant et de l'action choisie. Dans le cas déterministe, cette probabilité devient un delta de Kronecker. On suppose également que les coûts sont non négatifs pour éviter les cycles négatifs, et que l'état objectif est atteignable depuis n'importe quel état.

Politique optimale

Une **politique** π est définie comme un ensemble d'actions choisies dans chaque état. L'objectif est d'atteindre un état cible $s = d$, en partant d'un état initial $s_0 = k_0$ à $t = 0$, tout en minimisant le coût total attendu :

$$V_\pi(s_0 = k_0) = \mathbb{E}_{s_1, s_2, \dots} \left[\sum_{t=0}^{\infty} c(u(s_t) | s_t) \middle| s_0, \pi \right]$$

L'espérance est prise sur la séquence des états (variables aléatoires) jusqu'à atteindre l'état cible, qui est absorbant :

$$p(d | d, i) = 1 \quad \text{et} \quad c(d | d) = 0$$

La politique optimale π^* est alors définie par :

$$V^*(k_0) = \min_{\pi} \{V_\pi(k_0)\}$$

Algorithme d'itération sur les valeurs

Pour déterminer la meilleure politique, on peut utiliser l'algorithme d'itération sur les valeurs :

$$\begin{cases} \hat{V}(k) \leftarrow \min_{a \in U(k)} \left\{ c(a | k) + \sum_{k'} p(k' | k, a) \hat{V}(k') \right\}, & \text{si } k \neq d \\ \hat{V}(d) \leftarrow 0, & \text{où } d \text{ est l'état objectif} \end{cases}$$

En définissant a_t comme l'action associée à l'état observé $s_t = k_t$ au temps t , on peut heuristiquement dériver les conditions d'optimalité suivantes :

$$\begin{aligned}
V^*(k_0) &= \min_{(a_0, a_1, \dots)} \left\{ \mathbb{E}_{s_1, s_2, \dots} \left[\sum_{t=0}^{\infty} c(u(s_t) | s_t) \mid s_0 = k_0, \pi \right] \right\} \\
&= \min_{(a_0, a_1, \dots)} \left\{ \sum_{k_1, k_2, \dots} P(s_1 = k_1, s_2 = k_2, \dots \mid s_0 = k_0, \pi) \cdot \sum_{t=0}^{\infty} c(u(k_t) | k_t) \right\} \\
&= \min_{(a_0, a_1, \dots)} \left\{ \sum_{k_1, k_2, \dots} P(s_2 = k_2, s_3 = k_3, \dots \mid s_1 = k_1, \pi) \cdot P(s_1 = k_1 \mid s_0 = k_0, u(k_0) = a_0) \right. \\
&\quad \left. \cdot \left[c(a_0 \mid k_0) + \sum_{t=1}^{\infty} c(u(k_t) \mid k_t) \right] \right\} \\
&= \min_{a_0} \left\{ \sum_{k_1} P(s_1 = k_1 \mid s_0 = k_0, u(k_0) = a_0) \cdot \left[c(a_0 \mid k_0) \right. \right. \\
&\quad \left. \left. + \min_{(a_1, a_2, \dots)} \left\{ \sum_{k_2, k_3, \dots} P(s_2 = k_2, \dots \mid s_1 = k_1, \pi) \cdot \sum_{t=1}^{\infty} c(u(k_t) \mid k_t) \right\} \right] \right\} \\
&= \min_{a_0} \left\{ \sum_{k_1} p(k_1 \mid k_0, a_0) \cdot (c(a_0 \mid k_0) + V^*(k_1)) \right\} \\
&= \min_{a \in \mathcal{U}(k_0)} \left\{ \sum_{k_1} p(k_1 \mid k_0, a_0) \cdot (c(a_0 \mid k_0) + \hat{V}(k_1)) \right\}
\end{aligned}$$

Ce qui conduit à la formule d'itération sur les valeurs :

$$\begin{cases} \hat{V}(k) & \leftarrow \min_{a \in \mathcal{U}(k)} \left[c(a \mid k) + \sum_{k'=1}^n p(k' \mid k, a) \cdot \hat{V}(k') \right], \quad \text{pour } k \neq d \\ \hat{V}(d) & \leftarrow 0, \quad \text{où } d \text{ est l'état objectif} \end{cases}$$

Puisque le processus est sans mémoire (Markovien), on peut montrer que la politique optimale est stationnaire, c'est-à-dire qu'elle ne dépend pas du temps.

La meilleure action est donné, après convergence, par :

$$\operatorname{argmin}_{a \in U(k)} \left\{ \sum_{k'} p(k' \mid k, a) \left[c(a \mid k) + \hat{V}(k') \right] \right\}$$

Extensions possible

Dans les problèmes de décision de Markov (et RL), deux extensions classiques permettent de modéliser différents types de comportements temporels : le coût total actualisé et le coût moyen par unité de temps.

Le coût total actualisé consiste à minimiser la somme des coûts pondérés par un facteur d'actualisation $\gamma \in [0, 1]$, ce qui permet de donner plus d'importance aux coûts à court terme et d'assurer la convergence des sommes infinies. Ce critère est particulièrement adapté aux contextes où l'avenir est incertain ou où l'on souhaite privilégier des actions immédiates.

En revanche, le coût moyen par unité de temps (ou taux de coût) vise à minimiser le coût moyen à long terme, indépendamment du moment où il est encouru. Cette formulation est utile dans les systèmes en régime permanent, comme les chaînes de production ou les systèmes de maintenance, où l'on s'intéresse à la performance stable sur le long terme plutôt qu'à l'évolution transitoire. Ces deux critères permettent ainsi d'adapter la formulation du MDP à la nature du problème et aux objectifs visés.

Discuss the links with dynamic programming, the Q-value, reinforcement learning and Q-learning.

Lien avec la programmation dynamique

L'itération sur les valeurs repose sur l'équation de Bellman, commune à la programmation dynamique (DP). Dans les deux cas, la forme est récursive :

$$\min(\text{coût immédiat} + \text{coût futur}).$$

La différence majeure réside dans l'approche :

- En DP, l'algorithme est récursif avec un nombre fini d'états visités une seule fois, selon un ordre déterminé.
- En value iteration, l'algorithme est itératif, il revisite plusieurs fois les mêmes états jusqu'à convergence, qui dépend des valeurs initiales et d'un critère d'arrêt.

Enfin, en contexte stochastique, la valeur future n'est pas une simple valeur minimale mais une espérance : somme des valeurs possibles pondérées par leur probabilité.

Apprentissage par renforcement (RL)

Dans l'itération sur les valeurs, on suppose que l'environnement est connu, c'est-à-dire on suppose que les coûts, les états successeurs et les probabilités de transition sont connus, ce qui n'est pas toujours le cas (en particulier pour des problèmes de grande taille).

Avec l'apprentissage par renforcement, on apprend plutôt par l'expérience. On définit alors la **valeur Q** , qui représente le coût attendu lorsqu'on choisit une action a en état k et que l'on suit ensuite la politique π :

$$Q_\pi(k, a) = c(a | k) + \sum_{k'} p(k' | k, a) V_\pi(k')$$

Par conséquent :

$$V^*(k) = \min_{a \in U(k)} \{Q^*(k, a)\}$$

Ainsi, les conditions d'optimalité de Bellman exprimées en termes de Q-valeurs deviennent :

$$Q^*(k, a) = c(a | k) + \sum_{k'=1}^n p(k' | k, a) \min_{a' \in U(k')} \{Q^*(k', a')\} \quad (4)$$

On peut réécrire cela de la manière suivante :

$$\begin{aligned} Q^*(k, a) &= \sum_{k'=1}^n p(k' | k, a) \left[c(a | k) + \min_{a' \in U(k')} \{Q^*(k', a')\} \right] \\ &= \mathbb{E}_s \left[c(a | k) + \min_{a' \in U(k')} \{Q^*(s, a')\} \right] \end{aligned}$$

L'équivalent de l'itération sur les valeurs devient alors :

$$\begin{cases} \hat{Q}(k, a) \leftarrow c(a | k) + \sum_{k'=1}^n p(k' | k, a) \min_{a' \in U(k')} \{\hat{Q}(k', a')\}, & \text{si } k \neq d \\ \hat{Q}(d, a) \leftarrow 0, & \text{où } d \text{ est l'état objectif} \end{cases}$$

Pour ajuster la Q-valeur lorsque l'on essaie une action a en état k , qui mène à l'état k' avec un certain coût, on utilise une **approximation stochastique** :

$$\hat{Q}(k, a) \leftarrow \hat{Q}(k, a) + \alpha(t) \left(c(a | k) + \min_{a' \in U(k')} \{\hat{Q}(k', a')\} - \hat{Q}(k, a) \right), \quad \text{avec } k \neq d$$

Ceci est directement relié au **Q-learning**.

Cela provient du fait qu'une moyenne à un instant t peut être obtenue à l'aide de la formule suivante :

$$\hat{m} = \hat{m} + \alpha(t)(z(t) - \hat{m})$$

où z est une variable aléatoire. Si $\alpha(t) = \frac{1}{t}$, alors on calcule exactement la moyenne. Ce paramètre doit toujours satisfaire $0 < \alpha(t) < 1$, et doit décroître progressivement.

Preuve :

On souhaite calculer l'espérance empirique d'une variable aléatoire $z(t)$ à l'instant t . On a :

$$\mathbb{E}[z(t)] = \frac{1}{t} \sum_{\tau=1}^t z(\tau)$$

On peut réécrire cette somme comme suit :

$$\begin{aligned}\mathbb{E}[z(t)] &= \frac{1}{t} z(t) + \frac{1}{t} \sum_{\tau=1}^{t-1} z(\tau) \\ &= \frac{1}{t} z(t) + \frac{t-1}{t} \cdot \frac{1}{t-1} \sum_{\tau=1}^{t-1} z(\tau) \\ &= \frac{1}{t} z(t) + \frac{t-1}{t} \cdot \mathbb{E}[z(t-1)] \\ &= \mathbb{E}[z(t-1)] + \frac{1}{t} (z(t) - \mathbb{E}[z(t-1)])\end{aligned}$$

Ainsi, on obtient une mise à jour récursive de l'espérance :

$$\mathbb{E}[z(t)] = \mathbb{E}[z(t-1)] + \alpha(t) (z(t) - \mathbb{E}[z(t-1)])$$

où ici $\alpha(t) = \frac{1}{t}$ est un taux d'apprentissage décroissant.

Un mécanisme d'exploration doit également être mis en place afin de parcourir l'espace des états de manière aléatoire. Pour cela, on utilise généralement une stratégie dite ϵ -greedy, où le paramètre ϵ vérifie en général $0,9 < \epsilon < 1,0$.

Ce paramètre contrôle le compromis entre exploitation et exploration :

- avec une probabilité ϵ , on choisit l'action optimale actuelle (celle qui minimise la Q-valeur, i.e. *exploitation*)
- avec une probabilité $1 - \epsilon$, on choisit une action de manière aléatoire (i.e. *exploration*) afin de découvrir potentiellement de meilleures stratégies.

Une valeur élevée de ϵ favorise donc l'exploitation, tandis qu'une valeur plus faible augmente l'exploration.

Lorsque le nombre d'états devient trop grand, il est nécessaire d'utiliser une **approximation fonctionnelle** pour estimer les Q-valeurs, par exemple :

$$\hat{Q}(k, a) \approx f(x(k, a), w)$$

où x représente des caractéristiques (features) de l'état.

La règle de mise à jour devient alors une sorte de **descente de gradient** :

$$w_i \leftarrow w_i + \alpha(t) \left(c(a | k) + \min_{a' \in U(k')} \left\{ \hat{Q}(k', a') \right\} - \hat{Q}(k, a) \right) \cdot \frac{\partial f(x(k, a), w)}{\partial w_i}$$

Différence entre apprentissage par renforcement (RL) et Q-Learning

L'apprentissage par renforcement (RL) est un cadre général qui englobe toutes les méthodes où un agent apprend à optimiser son comportement via des interactions avec un environnement incertain. RL peut inclure diverses approches : méthodes basées sur des modèles (connaissance ou estimation des probabilités et coûts), méthodes sans modèle, apprentissage par politique, apprentissage par valeurs, etc.

Le Q-Learning est un algorithme particulier au sein de ce cadre RL. C'est une méthode d'apprentissage sans modèle (model-free) qui apprend directement la fonction de valeur $Q(k, a)$ optimale sans avoir besoin de connaître ni la dynamique de l'environnement ni les probabilités de transition.

Question 11 - Information Retrieval

Explain the probabilistic model of « information retrieval » and discuss the underlying assumptions, with the main equations. Moreover, how can we validate/assess an information retrieval system (precision, recall, F-measure).

How preprocess documents for information retrieval? [Bonus Question]

Avec une collection de documents et un ensemble d'utilisateurs souhaitant retrouver des documents correspondant à un concept donné, on soumet une requête exprimée à l'aide de mots ou de termes. Un système de recherche d'information renvoie alors les documents les plus liés à ce concept. Avant toute phase de recherche ou d'indexation, une étape de **prétraitement** des documents est nécessaire. Elle permet de transformer les textes bruts en représentations exploitables pour les algorithmes de recherche.

Voici les principales étapes du prétraitement :

- **Tokenisation** : découpage du texte en unités lexicales appelées *tokens* (mots, ponctuation, etc.). Par exemple, le texte « Bonjour le monde ! » sera transformé en la liste [Bonjour, le, monde, !].
- **Suppression des mots vides (stop words)** : suppression des mots fréquents et peu informatifs comme « le », « et », « est », etc. Ces mots n'apportent généralement pas de valeur discriminante pour la recherche de contenu pertinent.
- **Reconnaissance des entités nommées (NER - Named Entity Recognition)** : détection automatique des noms propres (personnes, lieux, organisations, dates, etc.). Cela permet d'ajouter un niveau sémantique important au document.
- **Racinement (stemming)** : réduction des mots à leur racine en supprimant les suffixes. Par exemple, « manger », « mangeons », « mangé » deviennent « mang ». Cela permet de regrouper différentes formes grammaticales d'un mot sous une même forme canonique.
- **Lemmatisation (optionnelle)** : alternative plus sophistiquée au stemming, qui réduit un mot à son lemme (forme de base) tout en tenant compte de son contexte grammatical. Par exemple, « suis » deviendra « être », ce que ne ferait pas le stemming.

Explain the probabilistic model of « information retrieval » and discuss the underlying assumptions, with the main equations.

Les méthodes probabilistes reposent sur des modèles statistiques, dans lesquels chaque utilisateur est représenté par un modèle probabiliste spécifique.

Dans ce cadre, le **modèle probabiliste** suppose qu'un document peut être soit pertinent ($R = 1$), soit non pertinent ($R = 0$). La notion de pertinence peut provenir d'un retour explicite de l'utilisateur (relevance feedback), ou être estimée automatiquement à partir d'un modèle d'espace vectoriel ou d'autres critères.

L'objectif est alors d'estimer la probabilité qu'un document soit pertinent vis-à-vis d'une requête donnée, et ainsi de classer les documents par ordre décroissant de cette probabilité.

Sur la base d'un classement, on sait que certains documents sont considérés comme pertinents ($R = 1$) pour un utilisateur u_k . On définit alors $\mathbb{P}(d = x \mid R = 1, u_k)$ comme la probabilité d'observer un document $d = x$ étant donné que ce document est pertinent pour l'utilisateur u_k . Cette probabilité peut être estimée facilement à l'aide d'un **modèle de recherche binaire d'indépendance**. Chaque document d_i est donc représenté par un vecteur binaire où $d_{ij} = 1$ si le mot w_j est présent dans le document d_i , et 0 sinon.

Cependant, dans le contexte de la recherche documentaire, on s'intéresse davantage à $\mathbb{P}(R = 1 \mid d = x, u_k)$, soit la probabilité qu'un document soit pertinent, étant donné son contenu. Cette probabilité doit être calculée pour chaque document.

Astuce : au lieu de calculer directement cette probabilité, on calcule le rapport de vraisemblance (ou odds) :

$$\lambda = \frac{\mathbb{P}(R = 1 \mid d = x, u_k)}{\mathbb{P}(R = 0 \mid d = x, u_k)}$$

Ce rapport fournit le même classement que la probabilité initiale car il s'agit d'une fonction monotone croissante. Plus λ est grand, plus le document est susceptible d'être pertinent.

Utilisation de la loi de Bayes :

$$\mathbb{P}(R = 1 \mid d = x, u_k) = \frac{\mathbb{P}(d = x \mid R = 1, u_k) \cdot \mathbb{P}(R = 1 \mid u_k)}{\mathbb{P}(d = x \mid u_k)} \quad \text{et de même pour } R = 0.$$

En supposant l'indépendance conditionnelle des mots :

$$\begin{aligned} \lambda &= \frac{\mathbb{P}(R = 1 \mid d = x, u_k)}{\mathbb{P}(R = 0 \mid d = x, u_k)} \\ &= \frac{\mathbb{P}(d = x \mid R = 1, u_k)}{\mathbb{P}(d = x \mid R = 0, u_k)} \cdot \frac{\mathbb{P}(R = 1 \mid u_k)}{\mathbb{P}(R = 0 \mid u_k)} \\ &= \underbrace{\frac{\mathbb{P}(R = 1 \mid u_k)}{\mathbb{P}(R = 0 \mid u_k)}}_{\text{a priori}} \cdot \underbrace{\prod_{n=1}^{n_w} \frac{\mathbb{P}(d_n = x_n \mid R = 1, u_k)}{\mathbb{P}(d_n = x_n \mid R = 0, u_k)}}_{\text{vraisemblance (a posteriori)}} \end{aligned}$$

Comme la probabilité à priori est fixe pour un utilisateur donné, λ est proportionnel au produit des termes de vraisemblance. Cela revient à utiliser un **classifieur naïf bayésien**.

Les probabilités à posteriori peuvent être facilement estimées par la proportion de documents contenant le mot w_n parmi les documents pertinents et non pertinents.

Remarque : Une hypothèse fondamentale du modèle présenté est que les mots sont supposés indépendants les uns des autres dans leur occurrence au sein d'un document — c'est l'hypothèse dite du sac de mots (bag-of-words). Cette simplification facilite les calculs, mais ignore les relations syntaxiques ou sémantiques entre les mots. De plus, seule la présence ou l'absence des mots est considérée, sans tenir compte de leur fréquence. Pour aller au-delà de cette hypothèse fondamentale, on peut recourir à des modèles plus sophistiqués, tels que :

- les modèles N-grammes, qui prennent en compte le contexte des $N - 1$ mots précédents, introduisant ainsi une dépendance locale entre les termes.
- les modèles de Poisson, qui modélisent non seulement la présence mais aussi la fréquence d'apparition des mots, en tenant compte de leur distribution dans les documents.

Moreover, how can we validate/assess an information retrieval system (precision, recall, F-measure).

Différentes métriques permettent d'évaluer le processus de recherche :

- **Précision (Precision)** : mesure le pourcentage de documents pertinents parmi les documents récupérés. La précision indique donc à quel point les documents récupérés sont pertinents.

$$Precision = \frac{TP}{TP + FP}$$

- **Rappel (Recall)** : mesure le pourcentage de documents pertinents récupérés parmi tous les documents pertinents disponibles. Le rappel indique donc à quel point les documents pertinents sont récupérés.

$$Recall = \frac{TP}{TP + FN}$$

- **F-mesure (F-measure)** : compromis entre précision et rappel. C'est la moyenne harmonique entre les deux mesures précédentes :

$$F\text{-measure} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Question 12 - Information Retrieval

Explain the basic vector model of « information retrieval », as well as two of its extensions (term reweighting and « latent semantic model »), with the main equations). Moreover, how can we validate/assess an information retrieval system (precision, recall, F-measure).

How preprocess documents for information retrieval? [Bonus Question]

Avec une collection de documents et un ensemble d'utilisateurs souhaitant retrouver des documents correspondant à un concept donné, on soumet une requête exprimée à l'aide de mots ou de termes. Un système de recherche d'information renvoie alors les documents les plus liés à ce concept. Avant toute phase de recherche ou d'indexation, une étape de **prétraitement** des documents est nécessaire. Elle permet de transformer les textes bruts en représentations exploitables pour les algorithmes de recherche.

Voici les principales étapes du prétraitement :

- **Tokenisation** : découpage du texte en unités lexicales appelées *tokens* (mots, ponctuation, etc.). Par exemple, le texte « Bonjour le monde ! » sera transformé en la liste [Bonjour, le, monde, !].
- **Suppression des mots vides (stop words)** : suppression des mots fréquents et peu informatifs comme « le », « et », « est », etc. Ces mots n'apportent généralement pas de valeur discriminante pour la recherche de contenu pertinent.
- **Reconnaissance des entités nommées (NER - Named Entity Recognition)** : détection automatique des noms propres (personnes, lieux, organisations, dates, etc.). Cela permet d'ajouter un niveau sémantique important au document.
- **Racinement (stemming)** : réduction des mots à leur racine en supprimant les suffixes. Par exemple, « manger », « mangeons », « mangé » deviennent « mang ». Cela permet de regrouper différentes formes grammaticales d'un mot sous une même forme canonique.
- **Lemmatisation (optionnelle)** : alternative plus sophistiquée au stemming, qui réduit un mot à son lemme (forme de base) tout en tenant compte de son contexte grammatical. Par exemple, « suis » deviendra « être », ce que ne ferait pas le stemming.

Explain the basic vector model of « information retrieval » with the main equations).

La technique du **modèle d'espace vectoriel** repose sur la représentation des documents, des requêtes, voire même des profils utilisateurs, sous forme de vecteurs dans un espace multidimensionnel. Les coordonnées de ces vecteurs correspondent aux mots, chaque élément du vecteur représentant la fréquence du mot dans le document ou la requête.

Le document j est noté d_j et f_{ij} représente la fréquence du mot w_i dans d_j . Le nombre total de mots est n_w , ce qui constitue également la dimension des vecteurs.

Cette représentation des documents est appelée **sac de mots (bag-of-words)**. L'ordre des mots dans un document n'a pas d'importance, et les vecteurs obtenus sont en général très *creux* (sparse) et de grande dimension.

Avec n_d le nombre de documents, la matrice terme-document est :

$$D \triangleq \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n_d} \\ f_{21} & f_{22} & \cdots & f_{2n_d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n_w 1} & f_{n_w 2} & \cdots & f_{n_w n_d} \end{bmatrix} \quad \text{où les lignes sont les mots et les colonnes sont les documents.}$$

Une requête est représentée par un vecteur q , également binaire (encodage one-hot). On souhaite retrouver les documents d_i qui correspondent le plus à la requête q . Pour cela, il nous faut une notion de similarité.

Similarité cosinus :

$$\text{sim}(q, d_i) = \cos(q, d_i) = \frac{q^\top d_i}{\|q\| \|d_i\|}$$

Cette métrique de similarité peut être appliquée à chaque document stocké dans D . La similarité cosinus est intéressante car elle compare l'angle entre deux vecteurs, et non leur distance ou leur norme. Et donc on compare bien les fréquences relatives des documents, et non absolue. La distance Euclidienne ne fonctionnerait pas bien car le nombre de mots dans la requête est beaucoup plus petit que ceux dans les documents.

Explain the extension "term reweighting".

Tous les mots dans les documents n'ont pas la même importance (par exemple : "le" , "la" , etc.). On souhaite donc intégrer une notion de pouvoir discriminant pour chaque mot.

Soit $P(w_i)$ la probabilité a priori qu'un mot w_i apparaisse dans un document. On définit alors la fréquence inverse de document (idf) comme :

$$idf_i = \log_2 \left(\frac{1}{P(w_i)} \right) = -\log_2(P(w_i))$$

Cette mesure quantifie l'importance globale d'un mot dans un corpus de documents.

- Plus idf_i est élevé, plus le mot est rare, et donc potentiellement informatif (ex. : "astrophysique").
- À l'inverse, une valeur faible (proche de 0) correspond à un mot très fréquent, généralement peu discriminant (ex. : "le", "la", "et").

Les valeurs de idf_i sont comprises entre 0 et $+\infty$. Un mot présent dans tous les documents (i.e. $P(w_i) = 1$) aura $idf_i = 0$, tandis qu'un mot rare, apparaissant dans très peu de documents, aura une IDF élevée.

La fréquence du terme (tf) est donnée par :

$$tf_{ij} = \frac{f_{ij}}{\sum_{i=0}^{n_w} f_{ij}}$$

où f_{ij} est la fréquence du terme w_i dans le document d_j , et la normalisation empêche les documents longs de dominer la mesure.

Le score $tf-idf$ est donc :

$$tf\text{-}idf_{ij} = idf_i \times tf_{ij}$$

Il est souvent utilisé avec la similarité cosinus, et peut également servir à redéfinir le vecteur requête \mathbf{q} comme :

$$q_i = \log_2 \left(\frac{1}{P(w_i)} \right) = -\log_2(P(w_i))$$

Chaque mot est alors pondéré par l'information apportée par sa présence dans la requête.

Explain the extension "latent semantic model".

Le modèle sémantique latent (*latent semantic model*, LSM) est une méthode utilisée pour capturer les relations sémantiques entre les mots à partir de leurs cooccurrences dans un ensemble de documents.

Deux mots sont considérés comme sémantiquement proches s'ils apparaissent fréquemment dans des contextes similaires (c'est-à-dire, s'ils ont une forte cooccurrence).

Pour extraire ces structures latentes, on utilise une méthode de réduction dimensionnelle, en particulier la **décomposition en valeurs singulières (SVD)**.

Remarque : Le *rang* d'une matrice correspond au nombre de colonnes (ou lignes) linéairement indépendantes. Dans le contexte terme-document, cela représente approximativement le nombre de thèmes ou concepts indépendants dans le corpus.

Dans ce cadre, on applique la SVD sur la matrice terme-document D pour en obtenir une version de rang réduit. Cela permet de :

- Réduire la dimensionnalité en regroupant les mots sémantiquement similaires (contextes proches),
- Rapprocher les documents partageant des thèmes communs.

En résumé, cette méthode permet de projeter la matrice terme-document D dans un sous-espace de dimension réduite, où chaque dimension correspond à un concept latent, facilitant ainsi la capture des relations sémantiques entre mots et documents.

Si D est la matrice terme-document (de taille $n_w \times n_d$), sa décomposition en valeurs singulières s'écrit :

$$D = U\Sigma V^T \quad \text{avec } U^T U = I, \quad V^T V = I$$

où :

- U contient la représentation des mots dans un espace latent,
- V^T contient la représentation des documents dans ce même espace,
- Σ est une matrice diagonale contenant les valeurs singulières, notée σ_i , classées par ordre décroissant.

On utilise cette décomposition spectrale, et non la forme $U^T \Lambda U$, car la matrice terme-document est rectangulaire. Cette décomposition peut être interprétée comme une double analyse en composantes principales (PCA) : une première appliquée aux lignes de D , et une seconde aux colonnes de D .

Pour obtenir une approximation de rang $m < \text{rang}(D)$, on conserve uniquement les m plus grandes valeurs singulières et on annule les suivantes :

$$\sigma_{m+1} = \sigma_{m+2} = \dots = \sigma_{\text{rang}(D)} = 0$$

On obtient alors une matrice réduite notée $\tilde{\Sigma}$, et l'approximation de D :

$$\tilde{D} = U \tilde{\Sigma} V^T$$

Cette nouvelle matrice \tilde{D} constitue la meilleure approximation de rang m . Autrement dit, aucune autre matrice de rang m n'est plus proche de D au sens de la norme de Frobenius :

$$\|\tilde{D} - D\|_F^2 = \sum_{i=1}^{n_w} \sum_{j=1}^{n_d} (\tilde{d}_{ij} - d_{ij})^2$$

Cette approximation permet donc de travailler dans un espace plus compact, tout en conservant l'essentiel de l'information sémantique. Elle améliore aussi les performances de calcul en réduisant la taille des données.

Les requêtes peuvent alors être projetées dans cet espace latent et comparées aux documents via la similarité cosinus :

$$\text{sim}(q, \tilde{d}_i) = \cos(q, \tilde{d}_i) = \frac{q^\top \tilde{d}_i}{\|q\| \|\tilde{d}_i\|}$$

Le principal inconvénient de cette méthode réside dans le choix du nombre de dimensions à conserver, qui n'est pas évident et peut fortement influencer les résultats. De plus, cette approche basée sur l'algèbre linéaire (notamment la SVD) n'est pas adaptée aux très grands corpus, car elle devient rapidement coûteuse en mémoire et en calcul pour des jeux de données de l'ordre de millions de documents.

Moreover, how can we validate/assess an information retrieval system (precision, recall, F-measure).

Différentes métriques permettent d'évaluer le processus de recherche :

- **Précision (Precision)** : mesure le pourcentage de documents pertinents parmi les documents récupérés. La précision indique donc à quel point les documents récupérés sont pertinents.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Rappel (Recall)** : mesure le pourcentage de documents pertinents récupérés parmi tous les documents pertinents disponibles. Le rappel indique donc à quel point les documents pertinents sont récupérés.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F-mesure (F-measure)** : compromis entre précision et rappel. C'est la moyenne harmonique entre les deux mesures précédentes :

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Question 13 - PageRank

Explain in detail the basic PageRank model for scoring web pages (without the random walk interpretation). Describe also its different extensions (personalization vector, etc). Then describe the « HITS » scoring model and its bibliometric interpretation.

Explain in detail the basic PageRank model for scoring web pages (without the random walk interpretation).

L'algorithme PageRank mesure le **prestige** dans un graphe orienté. Il exploite uniquement la structure du graphe, et non le contenu des pages. Dans ce graphe, les noeud représentent les documents et les arêtes les liens entre les documents.

L'objectif de l'algorithme PageRank est d'améliorer le classement des résultats des moteurs de recherche en exploitant la structure du graphe des hyperliens du Web. L'algorithme est calculé hors ligne, de manière préalable, afin de fournir un score d'importance à chaque page.

Chaque page a un score x_i qui est la moyenne pondérée des scores des pages pointant vers la page i . Soit w_{ij} le poids (non-négatif) entre les pages i et j , généralement 0 (pas de lien) ou 1 (lien existant). Tous les poids sont regroupés dans la matrice W , non symétrique dans le cas des graphes dirigés. On a donc :

$$x_i \propto \sum_{j=1}^n \frac{w_{ji}x_j}{w_j} \quad \text{avec} \quad w_j = \sum_{i=1}^n w_{ji}, \text{ le degré de sortie de la page } j$$

Une page avec un score élevé est donc une page pointée par de nombreuses pages ayant elles-mêmes des scores élevés. Une page est d'autant plus importante si elle est pointée par d'autres pages importantes qui possède peu de liens sortants.

Formulation du problème

PageRank est un algorithme fondé sur les concepts de chaînes de Markov. Il modélise la navigation sur le Web comme un processus stochastique, où chaque page correspond à un état, et chaque lien hypertexte à une transition entre états.

L'évolution de la distribution de probabilité sur les pages peut être décrite par l'équation suivante, en forme matricielle :

$$x(t) = P^T x(t-1)$$

où P est une matrice de transition stochastique de taille $n \times n$, et $x(t)$ est un vecteur de probabilité donnant la probabilité d'être sur chaque page (ou noeud) à l'instant t . $x(0)$ est généralement une distribution uniforme.

Méthode de la puissance pour PageRank

Les scores de PageRank correspondent à la **distribution stationnaire** de la chaîne de Markov, soit le **vecteur propre gauche** de P associé à la valeur propre 1 — ce qui revient à trouver le **vecteur propre droit** de P^T associé à cette même valeur propre.

On peut l'estimer efficacement via la méthode de la puissance, une méthode itérative définie par :

$$x(t) = P^T x(t-1)$$

À chaque itération, on normalise le vecteur x pour éviter qu'il ne diverge :

$$x(t) \leftarrow \frac{x(t)}{\|x(t)\|}$$

On répète cette opération jusqu'à convergence, c'est-à-dire jusqu'à ce que la variation entre deux vecteurs successifs soit inférieure à un certain seuil.

Remarque 1: Un vecteur propre droit d'une matrice A est le vecteur propre u tel que : $Au = \lambda u$. Un vecteur propre gauche d'une matrice A est le vecteur propre u tel que : $A^T u = \lambda u$. Autrement dit, un vecteur propre gauche d'une matrice A est simplement un vecteur propre droit de la matrice transposée A^T .

Remarque 2: La valeur propre maximale de P vaut 1 car P est une matrice de transition stochastique.

Remarque 3: Si le graphe n'est pas dirigé, les scores de PageRank deviennent proportionnels au nombre de connexions (ou degré) de chaque page, autrement dit au nombre de pages reliées à elle.

Describe also its different extensions (personalization vector, etc).

Téléportation

Un problème apparaît lorsqu'il existe des composantes disjointes dans le graphe, ou lorsque certaines pages (appelées *noeuds pendants*) ne possèdent aucun lien sortant. Cela peut bloquer la navigation du surfeur aléatoire et empêcher la convergence du vecteur de scores. Dans ce cas, la matrice P n'est plus stochastique.

Une solution consiste à autoriser un **saut aléatoire** vers n'importe quel noeud du graphe avec une probabilité non nulle. On introduit alors la **matrice de Google** G , définie par :

$$G = \alpha P + (1 - \alpha) \frac{1}{n} ee^T$$

où P est la matrice de transition (issue de la structure du graphe), e est un vecteur colonne rempli de 1, n est le nombre total de pages, et $0 < \alpha < 1$ est le facteur de probabilité de continuer la navigation via les liens existants (typiquement $\alpha \approx 0,85$). $\frac{1}{n}ee^T$ correspond à une distribution uniforme vers toutes les pages existantes du graphe.

Ce paramètre α représente donc la probabilité que le surfeur aléatoire suive un lien hypertexte existant. À l'inverse, avec une probabilité $1 - \alpha$, l'utilisateur effectue un saut aléatoire vers n'importe quelle page du Web, indépendamment des liens.

Les *noeuds pendants* (sans lien sortant) sont traités de plusieurs façons possibles :

- conversion en états absorbants,
- remplacement par une probabilité uniforme de saut vers toutes les pages.

On commence par transformer la matrice P afin de la rendre stochastique, ce qui est nécessaire pour appliquer correctement l'algorithme. Ensuite, bien que la matrice modifiée G (incluant le facteur de téléportation) soit dense, la méthode de la puissance peut être implémentée de manière efficace en exploitant la structure creuse de P , ce qui permet d'éviter de stocker ou manipuler G explicitement.

La matrice G ainsi construite est :

- **stochastique** (chaque ligne somme à 1),
- **irréductible** (chaque page est accessible depuis n'importe quelle autre),
- **apériodique** (absence de cycles bloquants).

Elle est donc **régulière**, ce qui garantit l'existence et l'unicité d'un vecteur propre positif associé à la valeur propre 1 (distribution stationnaire).

Les scores PageRank sont alors obtenus en déterminant le **vecteur propre gauche dominant** de G (ou vecteur propre droit dominant de G^T), via la **méthode de la puissance**:

$$\begin{cases} x^T(t+1) = x^T(t)G \\ x^T(t+1) \leftarrow \frac{x^T(t+1)}{\|x^T(t+1)\|_1} \end{cases}$$

Développée explicitement, l'itération devient :

$$\begin{aligned} x^T(k+1) &= x^T(k)G \\ &= \alpha x^T(k)P + \frac{1-\alpha}{n} x^T(k)ee^T \\ &= \alpha x^T(k)P + \frac{1-\alpha}{n} e^T \end{aligned}$$

où la dernière égalité découle du fait que $x^T(k)e = 1$ (normalisation). Le vecteur x est renormalisé à chaque itération pour que sa norme $\|x\|_1 = 1$.

Personnalisation

Comment peut-on favoriser certaines pages ?

Au lieu d'utiliser $\frac{ee^T}{n}$ dans la définition précédente, on peut utiliser ev^T avec $v > 0$ un vecteur de personnalisation tel que $v^T e = 1$. Il contient la probabilité *a priori* de téléportation vers chaque page. La matrice de Google devient alors :

$$G = \alpha P + (1 - \alpha)ev^T$$

Il y a une formule alternative qui est généralement un système creux d'équations linéaire :

$$(I - \alpha P)^T x = (1 - \alpha)v$$

Note: Si $\alpha = 1$ (pas de téléportation), on doit ajouter la contrainte que $e^T x = 1$ car $I - P$ est rang déficient.

Il existe également d'autres extensions, comme la possibilité que le vecteur v dépende de l'état courant, et ne soit donc pas constant ni défini uniquement *a priori*. D'autres variantes permettent aussi de modifier dynamiquement la matrice de transition P , par exemple en fonction du temps ou du contexte.

Then describe the « HITS » scoring model and its bibliometric interpretation.

HITS (*Hyperlink-Induced Topic Search*) est un autre modèle de ranking pour des graphes, tout comme PageRank.

Contrairement à PageRank qui donne un score unique à chaque page, HITS attribue deux scores à chaque page :

- **Authority (autorité)** : x_i^a mesure la valeur d'une page i comme source d'information fiable sur un sujet.
- **Hub (hub)** : x_i^h mesure la valeur d'une page i comme un répertoire ou une liste de liens vers des pages d'autorité.

Les hubs pointent vers de nombreuses pages d'autorité, ont peu de liens entrants, et devraient pointer vers de nombreuses autorités.

Les autorités ne pointent pas vers d'autres autorités, mais sont idéalement pointées par de bons hubs. En général, les autorités d'un même sujet sont en concurrence.

L'objectif de HITS est d'identifier à la fois les meilleurs experts (autorités) et les meilleurs guides (hubs) sur un sujet donné, ce qui est utile pour les recherches thématiques ou spécialisées.

Avec w_{ij} le poids du lien entre i et j , on a en forme matricielle :

$$x^a = \eta W^T x^h \quad \text{et} \quad x^h = \mu W x^a$$

D'où :

$$x^a = \eta \mu W^T W x^a \quad \text{et} \quad x^h = \eta \mu W W^T x^h$$

Les scores sont alors estimés par une procédure itérative (méthode de la puissance) équivalente au calcul des vecteurs propres de $W^T W$ (autorités) et $W W^T$ (hubs).

Lien avec la bibliométrie

L'algorithme HITS s'inspire de concepts fondamentaux en bibliométrie, la science qui étudie les citations et les relations entre documents scientifiques. Deux notions importantes sont :

- **Cocitation:** deux documents sont cocités s'ils sont tous deux cités par un même troisième document. Cela indique une relation thématique ou conceptuelle entre eux, car ils apparaissent ensemble dans la bibliographie d'une autre source.
- **Corréférence:** deux documents sont corréférencés s'ils citent tous deux un même troisième document. Cela reflète un lien par leur point commun, la référence partagée.

On a :

$$W^T W = D_{\text{in}} + C_{\text{cit}}, \quad W W^T = D_{\text{out}} + C_{\text{ref}}$$

avec :

- D_{in} : matrice diagonale des degrés entrants.
- D_{out} : matrice diagonale des degrés sortants.
- C_{cit} : matrice de cocitation.

- C_{ref} : matrice de corréférence.

Ainsi, la matrice des hubs est fortement liée à la matrice de corréférence, et la matrice des autorités à la matrice de cocitation.

HITS utilise implicitement ces notions bibliométriques pour identifier, dans un graphe de liens, les nœuds qui jouent le rôle d'autorités (sources d'information importantes) et de hubs (pages qui pointent vers ces sources).

Lien avec PCA

Il y a également un lien avec l'algorithme de feature extraction PCA. En effet, c'est exactement l'algorithme PCA non centré.

Question 14 - PageRank

Explain in detail the PageRank model for scoring web pages as well as its random walk (Markov chain) interpretation. Describe also its different extensions (personalization vector, etc).

Explain in detail the basic PageRank model for scoring web pages (without the random walk interpretation).

L'algorithme PageRank mesure le **prestige** dans un graphe orienté. Il exploite uniquement la structure du graphe, et non le contenu des pages. Dans ce graphe, les noeud représentent les documents et les arêtes les liens entre les documents.

L'objectif de l'algorithme PageRank est d'améliorer le classement des résultats des moteurs de recherche en exploitant la structure du graphe des hyperliens du Web. L'algorithme est calculé hors ligne, de manière préalable, afin de fournir un score d'importance à chaque page.

Chaque page a un score x_i qui est la moyenne pondérée des scores des pages pointant vers la page i . Soit w_{ij} le poids (non-négatif) entre les pages i et j , généralement 0 (pas de lien) ou 1 (lien existant). Tous les poids sont regroupés dans la matrice W , non symétrique dans le cas des graphes dirigés. On a donc :

$$x_i \propto \sum_{j=1}^n \frac{w_{ji}x_j}{w_j} \quad \text{avec } w_j = \sum_{i=1}^n w_{ji}, \text{ le degré de sortie de la page } j$$

Une page avec un score élevé est donc une page pointée par de nombreuses pages ayant elles-mêmes des scores élevés. Une page est d'autant plus importante si elle est pointée par d'autres pages importantes qui possède peu de liens sortants.

Formulation du problème

PageRank est un algorithme fondé sur les concepts de chaînes de Markov. Il modélise la navigation sur le Web comme un processus stochastique, où chaque page correspond à un état, et chaque lien hypertexte à une transition entre états.

L'évolution de la distribution de probabilité sur les pages peut être décrite par l'équation suivante, en forme matricielle :

$$x(t) = P^T x(t-1)$$

où P est une matrice de transition stochastique de taille $n \times n$, et $x(t)$ est un vecteur de probabilité donnant la probabilité d'être sur chaque page (ou nœud) à l'instant t .

Méthode de la puissance pour PageRank

Les scores de PageRank correspondent à la **distribution stationnaire** de la chaîne de Markov, soit le **vecteur propre gauche** de P associé à la valeur propre 1 — ce qui revient à trouver le **vecteur propre droit** de P^T associé à cette même valeur propre.

On peut l'estimer efficacement via la méthode de la puissance, une méthode itérative définie par :

$$x(t) = P^T x(t-1)$$

À chaque itération, on normalise le vecteur x pour éviter qu'il ne diverge :

$$x(t) \leftarrow \frac{x(t)}{\|x(t)\|}$$

On répète cette opération jusqu'à convergence, c'est-à-dire jusqu'à ce que la variation entre deux vecteurs successifs soit inférieure à un certain seuil.

Remarque 1: Un vecteur propre droit d'une matrice A est le vecteur propre u tel que : $Au = \lambda u$. Un vecteur propre gauche d'une matrice A est le vecteur propre u tel que : $A^T u = \lambda u$. Autrement dit, un vecteur propre gauche d'une matrice A est simplement un vecteur propre droit de la matrice transposée A^T .

Remarque 2: La valeur propre maximale de P vaut 1 car P est une matrice de transition stochastique.

Remarque 3: Si le graphe n'est pas dirigé, les scores de PageRank deviennent proportionnels au nombre de connexions (ou degré) de chaque page, autrement dit au nombre de pages reliées à elle.

Explain its random walk (Markov chain) interpretation.

L'algorithme PageRank peut être interprété comme le comportement d'un **surfeur aléatoire** qui navigue sur les pages d'un graphe (par exemple, des pages web reliées par des liens hypertextes). À chaque étape k , ce surfeur se trouve sur une page j et choisit de suivre l'un des liens sortants de cette page vers une page i , avec une probabilité proportionnelle au poids du lien.

Probabilité de transition

La probabilité de passer de la page j à la page i est donnée par :

$$P(\text{page}(k+1) = i \mid \text{page}(k) = j) = \frac{w_{ji}}{w_j}.$$

où :

- w_{ji} est le poids du lien allant de la page j vers la page i (généralement 1 si un lien existe, 0 sinon),
- $w_j = \sum_{i=1}^n w_{ji}$ est le nombre total de liens sortants depuis la page j .

Équation d'évolution

La probabilité d'être sur la page i à l'itération $k+1$ est donc donnée par :

$$\begin{aligned} x_i(k+1) &= P(\text{page}(k+1) = i) \\ &= \sum_{j=1}^n P(\text{page}(k+1) = i \mid \text{page}(k) = j) x_j(k) \\ &= \sum_{j=1}^n \frac{w_{ji}}{w_j} x_j(k) \end{aligned}$$

Si l'on note P la matrice de transition, dont les coefficients sont $p_{ij} = \frac{w_{ji}}{w_j}$, on peut réécrire l'équation sous forme matricielle :

$$x(k+1) = P^T x(k)$$

Distribution stationnaire

Lorsque le processus converge, on atteint une **distribution stationnaire** telle que :

$$x = P^T x$$

Le vecteur x est alors un **vecteur propre gauche** de la matrice P associé à la **valeur propre** 1 (équivalent au vecteur propre droit de P^T).

Méthode de la puissance

La solution x peut être approchée par la **méthode de la puissance**, via une procédure itérative :

$$x^{(k+1)} = P^T x^{(k)} \quad \text{avec une normalisation} \quad \|x^{(k+1)}\|_1 = 1$$

Conclusion

La valeur x_i représente la probabilité, à l'équilibre, que le surfeur se trouve sur la page i . L'algorithme PageRank classe donc les pages selon cette probabilité : plus une page est visitée fréquemment par le surfeur aléatoire, plus elle est jugée importante.

Describe also its different extensions (personalization vector, etc).

Téléportation

Un problème apparaît lorsqu'il existe des composantes disjointes dans le graphe, ou lorsque certaines pages (appelées *noeuds pendants*) ne possèdent aucun lien sortant. Cela peut bloquer la navigation du surfeur aléatoire et empêcher la convergence du vecteur de scores. Dans ce cas, la matrice P n'est plus stochastique.

Une solution consiste à autoriser un **saut aléatoire** vers n'importe quel noeud du graphe avec une probabilité non nulle. On introduit alors la **matrice de Google** G , définie par :

$$G = \alpha P + (1 - \alpha) \frac{1}{n} ee^T$$

où P est la matrice de transition (issue de la structure du graphe), e est un vecteur colonne rempli de 1, n est le nombre total de pages, et $0 < \alpha < 1$ est le facteur de probabilité de continuer la navigation via les liens existants (typiquement $\alpha \approx 0,85$).

Ce paramètre α représente donc la probabilité que le surfeur aléatoire suive un lien hypertexte existant. À l'inverse, avec une probabilité $1 - \alpha$, l'utilisateur effectue un saut aléatoire vers n'importe quelle page du Web, indépendamment des liens.

Les *noeuds pendants* (sans lien sortant) sont traités de plusieurs façons possibles :

- conversion en états absorbants,
- remplacement par une probabilité uniforme de saut vers toutes les pages.

On commence par transformer la matrice P afin de la rendre stochastique, ce qui est nécessaire pour appliquer correctement l'algorithme. Ensuite, bien que la matrice modifiée G (incluant le facteur de téléportation) soit dense, la méthode de la puissance peut être implémentée de manière efficace en exploitant la structure creuse de P , ce qui permet d'éviter de stocker ou manipuler G explicitement.

La matrice G ainsi construite est :

- **stochastique** (chaque ligne somme à 1),
- **irréductible** (chaque page est accessible depuis n'importe quelle autre),
- **apériodique** (absence de cycles bloquants).

Elle est donc **régulière**, ce qui garantit l'existence et l'unicité d'un vecteur propre positif associé à la valeur propre 1 (distribution stationnaire).

Les scores PageRank sont alors obtenus en déterminant le **vecteur propre gauche dominant** de G (ou vecteur propre droit dominant de G^T), via la **méthode de la puissance**:

$$\begin{cases} x^T(t+1) = x^T(t)G \\ x^T(t+1) \leftarrow \frac{x^T(t+1)}{\|x^T(t+1)\|_1} \end{cases}$$

Développée explicitement, l'itération devient:

$$\begin{aligned} x^T(k+1) &= x^T(k)G \\ &= \alpha x^T(k)P + \frac{1-\alpha}{n} x^T(k)ee^T \\ &= \alpha x^T(k)P + \frac{1-\alpha}{n} e^T \end{aligned}$$

où la dernière égalité découle du fait que $x^T(k)e = 1$ (normalisation). Le vecteur x est renormalisé à chaque itération pour que sa norme $\|x\|_1 = 1$.

Personnalisation

Comment peut-on favoriser certaines pages ?

Au lieu d'utiliser $\frac{ee^T}{n}$ dans la définition précédente, on peut utiliser ev^T avec $v > 0$ un vecteur de personnalisation tel que $v^T e = 1$. Il contient la probabilité *a priori* de téléportation vers chaque page. La matrice de Google devient alors :

$$G = \alpha P + (1 - \alpha)ev^T$$

Il y a une formule alternative qui est généralement un système creux d'équations linéaire :

$$(I - \alpha P)^T x = (1 - \alpha)v$$

Note: Si $\alpha = 1$ (pas de téléportation), on doit ajouter la contrainte que $e^T x = 1$ car $I - P$ est rang déficient.

Question 15 - Collab. Recom. Model

Describe in detail the basic model of collaborative recommendation, namely, the model based on k nearest neighbors. Also describe in detail the ItemRank model of collaborative recommendation (also called "random walk with restart"), inspired by PageRank. Finally, how do we assess a collaborative recommendation system?

Describe in detail the basic model of collaborative recommendation, namely, the model based on k nearest neighbors.

Pour la recommandation collaborative, on considère un ensemble d'individus et un ensemble d'objets (par exemple, des personnes et des films). Chaque individu est représenté par un vecteur \mathbf{v}_i , appelé **vecteur de profil**, dans l'espace des objets. Il contient un 1 s'il a acheté (ou interagi avec) l'objet, et 0 sinon.

La matrice de fréquence individu-objet W contient des éléments w_{ij} représentant soit des valeurs binaires (0 ou 1), soit le nombre de fois que l'individu i a acheté le produit j .

L'objectif de la recommandation collaborative est de recommander des objets aux individus en se basant sur les informations existantes (c'est-à-dire les achats ou interactions précédents).

Mesure de similarité

Il est nécessaire de calculer une **similarité** entre deux individus i et j à partir de leurs vecteurs de profil. Par exemple, on peut utiliser la **similarité cosinus** :

$$sim(i, j) = \frac{\mathbf{v}_i^T \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$$

D'autres techniques basées sur le nombre de valeurs correspondantes entre les vecteurs sont souvent utilisées. Une mesure typique de similarité est :

$$sim(i, j) = \frac{a}{a + b + c}$$

où :

- a est le nombre de correspondances où les deux vecteurs valent 1 (matchs positifs),
- b est le nombre de positions où i vaut 1 et j vaut 0,
- c est le nombre de positions où i vaut 0 et j vaut 1.

Les correspondances nulles (0-0) sont donc considérées comme non pertinentes dans ce calcul de similarité.

Algorithme des K plus proches voisins

En utilisant l'une de ces mesures de similarité, on peut appliquer un algorithme de type **K plus proches voisins (KNN)**. À partir des voisins les plus proches, c'est-à-dire les individus ayant les mêmes "goûts", on calcule des valeurs prédictives pour les objets non encore achetés et on recommande ceux ayant les scores les plus élevés.

La valeur prédictive de l'objet i pour l'individu p est calculée à partir des interactions de ses k plus proches voisins $q \in \mathcal{N}(p)$ avec cet objet :

$$pred(p, i) = \frac{\sum_{q \in \mathcal{N}_{\parallel}(p)} sim(p, q) \cdot w_{qi}}{\sum_{q \in \mathcal{N}_{\parallel}(p)} sim(p, q)}$$

où :

- w_{qi} est la fréquence d'interaction de l'individu q avec l'objet i ,
- $\mathcal{N}_{\parallel}(p)$ est l'ensemble des k plus proches voisins de la personne p ,
- $sim(p, q)$ est la similarité entre les individus p et q .

Ce modèle permet de recommander des objets que les voisins proches de l'utilisateur ont appréciés ou achetés fréquemment.

Also describe in detail the ItemRank model of collaborative recommendation (also called “random walk with restart”), inspired by PageRank.

Cet algorithme est basé sur le principe de PageRank, en y intégrant l'idée de **marche aléatoire avec redémarrage** (*Random Walk with Restart*).

On considère un **graphe biparti** constitué d'un ensemble d'utilisateurs et d'un ensemble d'objets. Un utilisateur est relié à un objet s'il l'a acheté.

On suppose qu'un **marcheur aléatoire** se déplace sur ce graphe. L'évolution de la probabilité $x_i(k)$ de se trouver sur le nœud i à l'itération k est donnée par :

$$x_i(k+1) = \alpha P^T x_i(k) + (1 - \alpha)v_i$$

où :

- P est la matrice de transition dérivée du graphe,
- v_i est un vecteur indiquant les objets achetés par l'utilisateur i : il contient $1/n_i$ pour chaque objet que l'utilisateur i a acheté (avec n_i le nombre total d'objets achetés), et 0 ailleurs,
- α est le facteur de redémarrage (avec $0 < \alpha < 1$).

Ainsi, à chaque itération, le marcheur aléatoire a une probabilité $(1 - \alpha)v_i$ d'être téléporté sur un objet déjà acheté par l'utilisateur i .

La solution stationnaire de cette équation, notée x_i , correspond aux scores de similarité associés à chaque objet pour l'utilisateur i :

$$x_i = (1 - \alpha)(I - \alpha P^T)^{-1}v_i$$

Ces scores permettent de recommander les objets les plus pertinents à l'utilisateur i .

Finally, how do we assess a collaborative recommendation system?

Différentes méthodes peuvent être utilisées pour évaluer les performances d'un système de recommandation.

Une approche courante consiste à **supprimer certains liens** (achats ou interactions) avant l'entraînement du modèle, afin de les utiliser comme ensemble de test. Cette méthode peut être combinée à une **validation croisée** pour améliorer la robustesse de l'évaluation. Les performances peuvent alors être mesurées à l'aide d'indicateurs tels que :

- **Précision (Precision)** : mesure la proportion d'objets recommandés qui sont réellement pertinents. Autrement dit, la précision indique à quel point les recommandations proposées sont justifiées.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Rappel (Recall)** : mesure la proportion des objets pertinents qui ont été effectivement recommandés. Le rappel indique donc la capacité du système à ne pas manquer des objets pertinents.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F-mesure (F-measure)** : compromis entre précision et rappel. C'est la moyenne harmonique entre les deux mesures précédentes :

$$F\text{-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Une bonne pratique consiste également à **comparer les performances du système** avec un modèle simple de fréquence maximale, qui consiste à toujours recommander les objets les plus populaires (par exemple, les films les plus vus).

D'autres propriétés qualitatives du système de recommandation peuvent également être évaluées, et donc incluses dans la fonction objective, telles que :

- la **surprise** : capacité à recommander des éléments inattendus ou nouveaux pour l'utilisateur,
- la **diversité** : variété des éléments recommandés, afin d'éviter la redondance dans les suggestions.

Question 16 - Collab. Recom. Model

Describe, but without detailing the EM algorithm applied to the complete likelihood function, the collaborative recommendation system based on the probabilistic latent classes model. Derive (mathematically) at least one update formula from a heuristic perspective (laws of probability). Moreover, how can we assess a collaborative recommendation system?

Describe, but without detailing the EM algorithm applied to the complete likelihood function, the collaborative recommendation system based on the probabilistic latent classes model. Derive (mathematically) at least one update formula from a heuristic perspective (laws of probability).

Dans ce modèle, on considère une variable aléatoire x représentant les individus (au nombre de m) et une variable y représentant les items (au nombre de n). On suppose l'existence d'une variable latente (non observée) z correspondant à des **classes d'intérêt** (au nombre de l) regroupant à la fois les utilisateurs et les items.

Le nombre de classes latentes l est donné *a priori*, et on suppose que x et y sont **indépendants conditionnellement à z** .

L'objectif est d'estimer la distribution de probabilité postérieure des items j pour un individu i , soit $P(y = j | x = i)$. Cela permet de recommander à un utilisateur les items les plus probables.

Pour la simplicité des notations, $P(x = x, y = y, z = k)$ sera parfois écrit comme $P(x, y, k)$.

En supposant l'indépendance conditionnelle de x et y sachant z , on a :

$$\begin{aligned} P(x, y) &= \sum_{k=1}^l P(x, y, k) \\ &= \sum_{k=1}^l P(x, y | k)P(k) \\ &= \sum_{k=1}^l P(x | k)P(y | k)P(k) \end{aligned}$$

Les variables latentes sont donc les classes $z = k$ avec $k = 1, \dots, l$.

Les paramètres du modèle sont définis comme les probabilités de masse des variables aléatoires discrètes :

$$\begin{cases} P(k) = P(z = k) \approx \hat{P}(k) & \text{probabilité à priori de la classe } k \\ P(x | k) = P(x = x | z = k) \approx \hat{P}(x | k) & \text{distribution des utilisateurs dans la classe } k \\ P(y | k) = P(y = y | z = k) \approx \hat{P}(y | k) & \text{distribution des objets dans la classe } k \end{cases}$$

Des N observations, on peut calculer $n(i, j)$ le nombre de fois qu'une transaction entre l'individu $x = i$ et l'item $y = j$ est observée dans la base de données.

On peut alors calculer la probabilité postérieure $P(y | x)$ comme suit :

$$\begin{aligned} P(y | x) &= \frac{P(x, y)}{P(x)} \\ &= \frac{\sum_{k=1}^l P(x | k)P(y | k)P(k)}{\sum_{j=1}^n \sum_{k'=1}^l P(x | z = k')P(y = j | z = k')P(z = k')} \\ &= \frac{\sum_{k=1}^l P(x | k)P(y | k)P(k)}{\sum_{k'=1}^l P(x | k')P(k')} \end{aligned}$$

Les paramètres du modèle sont estimés par **vraisemblance maximale**. Toutefois, cette maximisation directe peut s'avérer difficile. C'est pourquoi on utilise l'algorithme EM (Expectation-Maximization).

Algorithme EM (Expectation-Maximization)

L'algorithme EM est une méthode itérative utilisée pour **estimer les paramètres** d'un modèle en présence de **variables latentes** (non observées).

Le principe est de maximiser la log-vraisemblance complète des données (c'est-à-dire la vraisemblance des observations et des variables latentes) sans avoir directement accès aux variables latentes. On contourne cela en les estimant indirectement.

Chaque itération de l'algorithme EM comporte deux étapes :

- **Étape E (Espérance)** : On calcule l'espérance de la log-vraisemblance complète, notée

$$Q(\theta | \hat{\theta}^{(t)}) = \mathbb{E}_{z \sim P(z|x,y;\hat{\theta}^{(t)})} [\log P(x,y,z | \theta)]$$

Autrement dit, on prend l'espérance de la log-vraisemblance des données complètes (observées x, y et latentes z), conditionnellement aux observations et aux paramètres estimés à l'itération t . Cette étape revient à estimer la distribution des variables latentes et donc à estimer :

$$\hat{P}(z | x, y) \leftarrow \frac{\hat{P}(x | z)\hat{P}(y | z)\hat{P}(z)}{\sum_{k=1}^l \hat{P}(x | k)\hat{P}(y | k)\hat{P}(k)} \quad \text{voir logique à la page d'avant}$$

- **Étape M (Maximisation)** : En supposant les variables latentes z « connues » via l'estimation précédente, on maximise l'espérance calculée à l'étape E pour obtenir une nouvelle estimation des paramètres :

$$\hat{\theta}^{(t+1)} = \arg \max_{\theta} Q(\theta | \hat{\theta}^{(t)})$$

Cette étape permet d'estimer :

$$\begin{aligned} \hat{P}(z) &\leftarrow \frac{\sum_{x=1}^m \sum_{y=1}^n \hat{P}(z | x, y)n(x, y)}{\sum_{x=1}^m \sum_{y=1}^n n(x, y)} \quad \text{car } P(z) = \sum_{i=1}^m \sum_{j=1}^n P(z | x=i, y=j)P(x=i, y=j) \\ \hat{P}(x | z) &\leftarrow \frac{\sum_{y=1}^n \hat{P}(z | x, y)n(x, y)}{\sum_{x=1}^m \sum_{y=1}^n \hat{P}(z | x, y)n(x, y)} \quad \text{car } P(x | z) = \sum_{j=1}^n P(x, y=j | z) \\ \hat{P}(y | z) &\leftarrow \frac{\sum_{x=1}^m \hat{P}(z | x, y)n(x, y)}{\sum_{x=1}^m \sum_{y=1}^n \hat{P}(z | x, y)n(x, y)} \quad \text{car } P(y | z) = \sum_{i=1}^m P(x=i, y | z) \end{aligned}$$

Ce processus est répété jusqu'à convergence. Il peut être démontré que chaque itération **augmente la vraisemblance des données**, ce qui garantit une convergence vers un optimum local (mais pas forcément global).

Enfin, les prédictions sont fournies par l'estimateur $\hat{P}(y | x)$, obtenu en marginalisant sur les variables latentes selon les paramètres appris.

Finally, how do we assess a collaborative recommendation system?

Différentes méthodes peuvent être utilisées pour évaluer les performances d'un système de recommandation.

Une approche courante consiste à **supprimer certains liens** (achats ou interactions) avant l'entraînement du modèle, afin de les utiliser comme ensemble de test. Cette méthode peut être combinée à une **validation croisée** pour améliorer la robustesse de l'évaluation. Les performances peuvent alors être mesurées à l'aide d'indicateurs tels que :

- **Précision (Precision)** : mesure la proportion d'objets recommandés qui sont réellement pertinents. Autrement dit, la précision indique à quel point les recommandations proposées sont justifiées.

$$Precision = \frac{TP}{TP + FP}$$

- **Rappel (Recall)** : mesure la proportion des objets pertinents qui ont été effectivement recommandés. Le rappel indique donc la capacité du système à ne pas manquer des objets pertinents.

$$Recall = \frac{TP}{TP + FN}$$

- **F-mesure (F-measure)** : compromis entre précision et rappel. C'est la moyenne harmonique entre les deux mesures précédentes :

$$F\text{-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Une bonne pratique consiste également à **comparer les performances du système** avec un modèle simple de fréquence maximale, qui consiste à toujours recommander les objets les plus populaires (par exemple, les films les plus vus).

D'autres propriétés qualitatives du système de recommandation peuvent également être évaluées, et donc incluses dans la fonction objective, telles que :

- la **surprise** : capacité à recommander des éléments inattendus ou nouveaux pour l'utilisateur,
- la **diversité** : variété des éléments recommandés, afin d'éviter la redondance dans les suggestions.

Question 17 - Collab. Recom. Model

Describe in detail the basic model of collaborative recommendation, namely, the model based on k nearest neighbors. Also explain in detail the non-negative matrix factorization techniques for collaborative recommendation based on ratings. Moreover, how can we assess a collaborative recommendation system?

Describe in detail the basic model of collaborative recommendation, namely, the model based on k nearest neighbors.

Pour la recommandation collaborative, on considère un ensemble d'individus et un ensemble d'objets (par exemple, des personnes et des films). Chaque individu est représenté par un vecteur \mathbf{v}_i , appelé **vecteur de profil**, dans l'espace des objets. Il contient un 1 s'il a acheté (ou interagi avec) l'objet, et 0 sinon.

La matrice de fréquence individu-objet W contient des éléments w_{ij} représentant soit des valeurs binaires (0 ou 1), soit le nombre de fois que l'individu i a acheté le produit j .

L'objectif de la recommandation collaborative est de recommander des objets aux individus en se basant sur les informations existantes (c'est-à-dire les achats ou interactions précédents).

Mesure de similarité

Il est nécessaire de calculer une **similarité** entre deux individus i et j à partir de leurs vecteurs de profil. Par exemple, on peut utiliser la **similarité cosinus** :

$$sim(i, j) = \frac{\mathbf{v}_i^T \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$$

D'autres techniques basées sur le nombre de valeurs correspondantes entre les vecteurs sont souvent utilisées. Une mesure typique de similarité est :

$$sim(i, j) = \frac{a}{a + b + c}$$

où :

- a est le nombre de correspondances où les deux vecteurs valent 1 (matchs positifs),
- b est le nombre de positions où i vaut 1 et j vaut 0,
- c est le nombre de positions où i vaut 0 et j vaut 1.

Les correspondances nulles (0-0) sont donc considérées comme non pertinentes dans ce calcul de similarité.

Algorithme des K plus proches voisins

En utilisant l'une de ces mesures de similarité, on peut appliquer un algorithme de type **K plus proches voisins (KNN)**. À partir des voisins les plus proches, c'est-à-dire les individus ayant les mêmes "goûts", on calcule des valeurs prédictives pour les objets non encore achetés et on recommande ceux ayant les scores les plus élevés.

La valeur prédictive de l'objet i pour l'individu p est calculée à partir des interactions de ses k plus proches voisins $q \in \mathcal{N}(p)$ avec cet objet :

$$pred(p, i) = \frac{\sum_{q \in \mathcal{N}_{\parallel}(p)} sim(p, q) \cdot w_{qi}}{\sum_{q \in \mathcal{N}_{\parallel}(p)} sim(p, q)}$$

où :

- w_{qi} est la fréquence d'interaction de l'individu q avec l'objet i ,
- $\mathcal{N}_{\parallel}(p)$ est l'ensemble des k plus proches voisins de la personne p ,
- $sim(p, q)$ est la similarité entre les individus p et q .

Ce modèle permet de recommander des objets que les voisins proches de l'utilisateur ont appréciés ou achetés fréquemment.

Also explain in detail the non-negative matrix factorization techniques for collaborative recommendation based on ratings.

On considère une matrice $W \geq 0$, représentant les interactions entre utilisateurs et items. Les éléments de cette matrice peuvent être approximés par le produit scalaire de deux vecteurs latents : u_i , vecteur latent représentant les préférences de l'utilisateur i , et v_j , vecteur latent représentant les attributs de l'item j .

On impose une contrainte de non-négativité sur ces vecteurs (contrairement à une décomposition en valeurs singulières classique). On suppose également que l'espace latent est de faible dimension. En notant U et V les matrices contenant respectivement les vecteurs u_i et v_j , on a :

$$w_{ij} \approx u_i^T v_j \quad \text{ou, en forme matricielle,} \quad W \approx UV^T$$

Cela mène au problème d'optimisation non-convexe (et difficile à résoudre) suivant :

$$\min_{U,V} \{ \|W - UV^T\|_F^2 \} \quad \text{sous la contrainte} \quad U, V \geq 0 \quad \text{où} \quad \|\cdot\|_F \quad \text{désigne la norme de Frobenius.}$$

Les contraintes $U, V \geq 0$ sont assez naturelles, car un poids représente une contribution entre un utilisateur et un item. Autoriser des poids négatifs reviendrait à permettre que certaines contributions s'annulent, ce qui a peu de sens dans le cadre des systèmes de recommandation collaboratifs, où les interactions sont généralement interprétées comme positives ou nulles, mais rarement comme "négativement contributives".

Méthode des moindres carrés alternés

Une approche populaire consiste à utiliser une méthode d'optimisation alternée :

$$\min_U \|W - UV^T\|_F^2 \quad \text{avec} \quad U \geq 0 \text{ et } V \text{ fixé,}$$

$$\min_V \|W - UV^T\|_F^2 \quad \text{avec} \quad V \geq 0 \text{ et } U \text{ fixé}$$

Ce processus est répété jusqu'à convergence.

Méthode pratique sans garantie théorique

Une autre technique, encore plus populaire, consiste à résoudre les mêmes problèmes sans contrainte explicite, puis à tronquer les éléments négatifs :

$$\min_U \|W - UV^T\|_F^2 \quad \text{et ensuite fixer à zéro les éléments négatifs de } U$$

$$\min_V \|W - UV^T\|_F^2 \quad \text{et ensuite fixer à zéro les éléments négatifs de } V$$

Cette méthode fonctionne bien empiriquement, même si aucune preuve de convergence n'est garantie.

Cas avec valeurs manquantes

Toutefois, ces méthodes supposent que la matrice W n'est pas trop creuse (qu'elle ne contient pas trop de valeurs manquantes). En présence de nombreuses valeurs manquantes, les prédictions peuvent être biaisées.

Une solution consiste à exclure les éléments manquants de la fonction objectif. En notant E l'ensemble des couples (i, j) pour lesquels une interaction existe, on résout :

$$\min_{U,V \geq 0} \sum_{(i,j) \in E} (W_{ij} - u_i^T v_j)^2$$

Cette version du problème peut être optimisée à l'aide de la **descente de gradient stochastique**.

Finally, how do we assess a collaborative recommendation system?

Différentes méthodes peuvent être utilisées pour évaluer les performances d'un système de recommandation.

Une approche courante consiste à **supprimer certains liens** (achats ou interactions) avant l'entraînement du modèle, afin de les utiliser comme ensemble de test. Cette méthode peut être combinée à une **validation croisée** pour améliorer la robustesse de l'évaluation. Les performances peuvent alors être mesurées à l'aide d'indicateurs tels que :

- **Précision (Precision)** : mesure la proportion d'objets recommandés qui sont réellement pertinents. Autrement dit, la précision indique à quel point les recommandations proposées sont justifiées.

$$Precision = \frac{TP}{TP + FP}$$

- **Rappel (Recall)** : mesure la proportion des objets pertinents qui ont été effectivement recommandés. Le rappel indique donc la capacité du système à ne pas manquer des objets pertinents.

$$Recall = \frac{TP}{TP + FN}$$

- **F-mesure (F-measure)** : compromis entre précision et rappel. C'est la moyenne harmonique entre les deux mesures précédentes :

$$F\text{-measure} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Une bonne pratique consiste également à **comparer les performances du système** avec un modèle simple de fréquence maximale, qui consiste à toujours recommander les objets les plus populaires (par exemple, les films les plus vus).

D'autres propriétés qualitatives du système de recommandation peuvent également être évaluées, et donc incluses dans la fonction objective, telles que :

- la **surprise** : capacité à recommander des éléments inattendus ou nouveaux pour l'utilisateur,
- la **diversité** : variété des éléments recommandés, afin d'éviter la redondance dans les suggestions.

Question 18 - Markov Chain (Evol. Eq.)

Describe what is a Markov chain and derive in detail (mathematically) its evolution equation. Also discuss some applications without going into the mathematical details.

Describe what is a Markov chain

Une chaîne de Markov est un modèle de processus stochastiques à temps discret et à états discrets. On considère un ensemble fini d'états $S = \{1, 2, \dots, n\}$, et $s_t = k$ signifie que le processus se trouve dans l'état k à l'instant t .

On définit une matrice de transition P avec des éléments $p_{ij} = \mathbb{P}(s_{t+1} = j \mid s_t = i)$. On suppose que la probabilité de transition d'un état à un autre dépend uniquement de l'état courant (propriété de Markov). La matrice P représente donc les probabilités de transition en un seul pas, et est une matrice stochastique, c'est-à-dire que la somme de chaque ligne vaut 1.

Transition à deux pas

On peut calculer les probabilités de transition en deux étapes de la manière suivante :

$$\begin{aligned}\mathbb{P}(s_{t+2} = j \mid s_t = i) &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j, s_{t+1} = k \mid s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j \mid s_t = i, s_{t+1} = k) \cdot \mathbb{P}(s_{t+1} = k \mid s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j \mid s_{t+1} = k) \cdot \mathbb{P}(s_{t+1} = k \mid s_t = i) \quad \text{par la propriété de Markov} \\ &= \sum_{k=1}^n p_{kj} \cdot p_{ik} \\ &= [P^2]_{ij}\end{aligned}$$

Transition en τ étapes

Par récurrence, on obtient que P^τ est la matrice de transition à τ étapes, avec :

$$p_{ij}^{(\tau)} = \mathbb{P}(s_{t+\tau} = j \mid s_t = i)$$

Propriétés des chaînes de Markov

Une chaîne de Markov est dite **irréductible** si, à partir de n'importe quel état, il est possible d'atteindre tous les autres états, éventuellement en plusieurs étapes.

De plus, une chaîne de Markov est dite **régulière** si une certaine puissance de sa matrice de transition contient uniquement des éléments strictement positifs (non nuls).

Lorsqu'une chaîne de Markov est régulière, on peut montrer que les puissances successives de la matrice de transition, notées P^t , tendent vers une matrice dont toutes les lignes sont identiques.

Cela implique que la distribution limite des probabilités des états ne dépend pas de l'état initial : le système perd sa mémoire avec le temps.

Le vecteur stationnaire p est alors le vecteur propre gauche de la matrice P , associé à la valeur propre 1, normalisé de façon à être une distribution de probabilités :

$$\begin{aligned}\pi &= \lim_{t \rightarrow \infty} x(t) \\ &= \lim_{t \rightarrow \infty} P^T x(t-1) \\ &= P^T \lim_{t \rightarrow \infty} x(t-1) \\ &= P^T \pi\end{aligned}$$

Derive in detail (mathematically) its evolution equation.

On définit $x(t)$, un vecteur colonne contenant la distribution de probabilité d'être dans chacun des états à l'instant t :

$$\begin{aligned} x_i(t) &= \mathbb{P}(s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_t = i, s_{t-1} = k) \\ &= \sum_{k=1}^n \mathbb{P}(s_t = i | s_{t-1} = k) \cdot \mathbb{P}(s_{t-1} = k) \\ &= \sum_{k=1}^n p_{ki} \cdot x_k(t-1) \end{aligned}$$

En notation matricielle, cela s'écrit :

$$x(t) = P^T x(t-1) \Rightarrow x(t) = (P^T)^t x(0)$$

Cette relation est appelée **l'équation d'évolution temporelle** de la chaîne de Markov, avec $x(0)$ représentant la distribution initiale des probabilités.

On peut trouver $x_j(t)$, la probabilité d'être dans l'état j au temps t avec la formule suivante:

$$x_j(t) = x(t)^T e_j = x(0)^T P^t e_j$$

Si l'on commence dans l'état i , alors la condition initiale est $x(0) = e_i$, où e_i est le vecteur unitaire (composé de zéros sauf un 1 à la position i).

Ainsi, la probabilité d'être dans l'état j au temps t en partant de i est donnée par :

$$x_j(t | s_0 = i) = x_{j|i}(t) = (e_i)^T P^t e_j = (e_j)^T (P^T)^t e_i$$

Also discuss some applications without going into the mathematical details.

Les chaînes de Markov trouvent des applications dans de nombreux domaines, notamment le filtrage collaboratif et le marketing.

Filtrage collaboratif: Un exemple d'application est l'algorithme *ItemRank*, utilisé pour recommander des articles ou produits à un utilisateur en se basant sur les transitions probables entre éléments. La navigation d'un utilisateur entre articles est modélisée comme un processus stochastique, dans lequel chaque item correspond à un état de la chaîne.

Cet exemple est défini en détail dans la Question 14.

Marketing: Dans un contexte de marketing relationnel, on peut segmenter les clients en différents clusters à l'aide du modèle RFM (Récence, Fréquence, Valeur Monétaire). Chaque cluster représente un état d'une chaîne de Markov. Le dernier cluster (souvent le n -ième) représente les clients perdus ; cet état est alors modélisé comme absorbant (on ne peut plus en sortir).

En observant les transitions mensuelles des clients entre clusters, on peut estimer les probabilités de transition. En associant un bénéfice moyen à chaque cluster, on peut alors calculer le **profit espéré généré par un client au cours de son cycle de vie**, également appelé *customer lifetime value* (CLV). Cette approche permet de mieux orienter les efforts de rétention client.

Cet exemple est défini en détail dans la Question 20.

Autres domaines d'application:

- **Reconnaissance vocale et traitement du langage** : les modèles de Markov (souvent cachés – HMM) permettent de représenter la structure probabiliste des séquences de sons ou de mots.
- **Modélisation du comportement utilisateur** : dans les jeux vidéo, ou sur des plateformes en ligne, pour anticiper les actions futures d'un joueur ou d'un internaute.
- **Finance** : modélisation de l'évolution de la solvabilité d'un emprunteur ou du passage entre différents états de crédit.

Question 19 - Markov Chain (# Of Visits)

Describe what is a Markov chain and derive in detail (mathematically) how to compute the expected number of visits to each state in the case of absorbing Markov chains. Also discuss some game/finance application.

Describe what is a Markov chain

Une chaîne de Markov est un modèle de processus stochastiques à temps discret et à états discrets. On considère un ensemble fini d'états $S = \{1, 2, \dots, n\}$, et $s_t = k$ signifie que le processus se trouve dans l'état k à l'instant t .

On définit une matrice de transition P avec des éléments $p_{ij} = \mathbb{P}(s_{t+1} = j \mid s_t = i)$. On suppose que la probabilité de transition d'un état à un autre dépend uniquement de l'état courant (propriété de Markov). La matrice P représente donc les probabilités de transition en un seul pas, et est une matrice stochastique, c'est-à-dire que la somme de chaque ligne vaut 1.

Transition à deux pas

On peut calculer les probabilités de transition en deux étapes de la manière suivante :

$$\begin{aligned}\mathbb{P}(s_{t+2} = j \mid s_t = i) &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j, s_{t+1} = k \mid s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j \mid s_t = i, s_{t+1} = k) \cdot \mathbb{P}(s_{t+1} = k \mid s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j \mid s_{t+1} = k) \cdot \mathbb{P}(s_{t+1} = k \mid s_t = i) \quad \text{par la propriété de Markov} \\ &= \sum_{k=1}^n p_{kj} \cdot p_{ik} \\ &= [P^2]_{ij}\end{aligned}$$

Transition en τ étapes

Par récurrence, on obtient que P^τ est la matrice de transition à τ étapes, avec :

$$p_{ij}^{(\tau)} = \mathbb{P}(s_{t+\tau} = j \mid s_t = i)$$

Propriétés des chaînes de Markov

Une chaîne de Markov est dite **irréductible** si, à partir de n'importe quel état, il est possible d'atteindre tous les autres états, éventuellement en plusieurs étapes.

De plus, une chaîne de Markov est dite **régulière** si une certaine puissance de sa matrice de transition contient uniquement des éléments strictement positifs (non nuls).

Lorsqu'une chaîne de Markov est régulière, on peut montrer que les puissances successives de la matrice de transition, notées P^t , tendent vers une matrice dont toutes les lignes sont identiques.

Cela implique que la distribution limite des probabilités des états ne dépend pas de l'état initial : le système perd sa mémoire avec le temps.

Le vecteur stationnaire p est alors le vecteur propre gauche de la matrice P , associé à la valeur propre 1, normalisé de façon à être une distribution de probabilités :

$$\begin{aligned}\pi &= \lim_{t \rightarrow \infty} x(t) \\ &= \lim_{t \rightarrow \infty} P^T x(t-1) \\ &= P^T \lim_{t \rightarrow \infty} x(t-1) \\ &= P^T \pi\end{aligned}$$

Derive in detail (mathematically) how to compute the expected number of visits to each state in the case of absorbing Markov chains.

Un état est dit **absorbant** s'il est impossible d'en sortir, c'est-à-dire si $p_{ii} = 1$. Les autres états sont appelés **transitoires**.

On peut réécrire la matrice de transition P sous forme canonique :

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$$

où :

- Q est la matrice de transition entre les états transitoires ;
- R est la matrice des probabilités de transition entre états transitoires et absorbants ;
- 0 est une matrice nulle ;
- I est la matrice identité correspondant aux états absorbants.

Les matrices Q et R sont dites **sub-stochastiques** (la somme des lignes est inférieure ou égale à 1).

Pour calculer P^t , on peut simplement se concentrer sur la puissance Q^t . Comme Q est sub-stochastique, alors $Q^t \rightarrow 0$ lorsque $t \rightarrow \infty$.

On définit alors la **matrice fondamentale** de la chaîne absorbante par :

$$\begin{aligned} N &= \sum_{t=0}^{\infty} Q^t \\ &= I + Q + Q^2 + Q^3 + \dots \\ &= (I - Q)^{-1} \quad \text{car c'est une suite géométrique} \end{aligned}$$

Les éléments n_{ij} de cette matrice, pour i, j correspondant à des états transitoires, s'interprètent comme :

$$\begin{aligned} n_{ij} &= e_i^T N e_j \\ &= e_i^T \left(\sum_{t=0}^{\infty} Q^t \right) e_j \\ &= \sum_{t=0}^{\infty} e_i^T Q^t e_j \\ &= \sum_{t=0}^{\infty} x_{j|i}(t) \quad \text{car, dans ce contexte, } Q^t = P^t \text{ puisque l'on considère uniquement les états transitoires} \\ &= \mathbb{E}[\text{nombre de visites en l'état } j \mid \text{départ en l'état } i] \end{aligned}$$

Cela signifie que n_{ij} est le nombre espéré de passages par l'état transitoire j lorsque l'on part de l'état i . Ainsi, le **nombre total de visites avant absorption** en partant de chaque état est donné par $n = Ne$ où e est un vecteur colonne de 1.

Also discuss some game/finance application.

Les chaînes de Markov trouvent des applications dans de nombreux domaines, notamment le filtrage collaboratif et le marketing.

Filtrage collaboratif: Un exemple d'application est l'algorithme *ItemRank*, utilisé pour recommander des articles ou produits à un utilisateur en se basant sur les transitions probables entre éléments. La navigation d'un utilisateur entre articles est modélisée comme un processus stochastique, dans lequel chaque item correspond à un état de la chaîne.

Cet exemple est défini en détail dans la Question 14.

Marketing: Dans un contexte de marketing relationnel, on peut segmenter les clients en différents clusters à l'aide du modèle RFM (Récence, Fréquence, Valeur Monétaire). Chaque cluster représente un état d'une chaîne

de Markov. Le dernier cluster (souvent le n -ième) représente les clients perdus ; cet état est alors modélisé comme absorbant (on ne peut plus en sortir).

En observant les transitions mensuelles des clients entre clusters, on peut estimer les probabilités de transition. En associant un bénéfice moyen à chaque cluster, on peut alors calculer le **profit espéré généré par un client au cours de son cycle de vie**, également appelé *customer lifetime value* (CLV). Cette approche permet de mieux orienter les efforts de rétention client.

Cet exemple est défini en détail dans la Question 20.

Autres domaines d'application:

- **Reconnaissance vocale et traitement du langage** : les modèles de Markov (souvent cachés – HMM) permettent de représenter la structure probabiliste des séquences de sons ou de mots.
- **Modélisation du comportement utilisateur** : dans les jeux vidéo, ou sur des plateformes en ligne, pour anticiper les actions futures d'un joueur ou d'un internaute.
- **Finance** : modélisation de l'évolution de la solvabilité d'un emprunteur ou du passage entre différents états de crédit.

Question 20 - Markov Chain (Absorb. Probab.)

Describe what is an absorbing Markov chain and derive in detail (mathematically) how to compute the absorbing probabilities (probability of being absorbed in an absorbing state). Also discuss some game/finance application.

Describe what is an absorbing Markov chain.

Une chaîne de Markov est un modèle de processus stochastiques à temps discret et à états discrets. On considère un ensemble fini d'états $S = \{1, 2, \dots, n\}$, et $s_t = k$ signifie que le processus se trouve dans l'état k à l'instant t .

On définit une matrice de transition P avec des éléments $p_{ij} = \mathbb{P}(s_{t+1} = j \mid s_t = i)$. On suppose que la probabilité de transition d'un état à un autre dépend uniquement de l'état courant (propriété de Markov). La matrice P représente donc les probabilités de transition en un seul pas, et est une matrice stochastique, c'est-à-dire que la somme de chaque ligne vaut 1.

Transition à deux pas

On peut calculer les probabilités de transition en deux étapes de la manière suivante :

$$\begin{aligned}\mathbb{P}(s_{t+2} = j \mid s_t = i) &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j, s_{t+1} = k \mid s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j \mid s_t = i, s_{t+1} = k) \cdot \mathbb{P}(s_{t+1} = k \mid s_t = i) \\ &= \sum_{k=1}^n \mathbb{P}(s_{t+2} = j \mid s_{t+1} = k) \cdot \mathbb{P}(s_{t+1} = k \mid s_t = i) \quad \text{par la propriété de Markov} \\ &= \sum_{k=1}^n p_{kj} \cdot p_{ik} \\ &= [P^2]_{ij}\end{aligned}$$

Transition en τ étapes

Par récurrence, on obtient que P^τ est la matrice de transition à τ étapes, avec :

$$p_{ij}^{(\tau)} = \mathbb{P}(s_{t+\tau} = j \mid s_t = i)$$

Propriétés des chaînes de Markov

Une chaîne de Markov est dite **irréductible** si, à partir de n'importe quel état, il est possible d'atteindre tous les autres états, éventuellement en plusieurs étapes.

De plus, une chaîne de Markov est dite **régulière** si une certaine puissance de sa matrice de transition contient uniquement des éléments strictement positifs (non nuls).

Lorsqu'une chaîne de Markov est régulière, on peut montrer que les puissances successives de la matrice de transition, notées P^t , tendent vers une matrice dont toutes les lignes sont identiques.

Cela implique que la distribution limite des probabilités des états ne dépend pas de l'état initial : le système perd sa mémoire avec le temps.

Le vecteur stationnaire p est alors le vecteur propre gauche de la matrice P , associé à la valeur propre 1, normalisé de façon à être une distribution de probabilités :

$$\begin{aligned}\pi &= \lim_{t \rightarrow \infty} x(t) \\ &= \lim_{t \rightarrow \infty} P^T x(t-1) \\ &= P^T \lim_{t \rightarrow \infty} x(t-1) \\ &= P^T \pi\end{aligned}$$

Chaîne de Markov absorbante

Une chaîne de Markov absorbante est une chaîne de Markov particulière dans laquelle il existe au moins un état absorbant. Un état absorbant est un état s_i tel que :

$$p_{ii} = 1 \quad \text{et} \quad p_{ij} = 0 \quad \forall j \neq i.$$

Autrement dit, une fois que le processus atteint un état absorbant, il y reste pour toujours.

Un état non absorbant est appelé transitoire s'il est possible de le quitter pour atteindre un état absorbant.

Une chaîne est dite absorbante si :

- Il existe au moins un état absorbant.
- À partir de tout état transitoire, il existe une probabilité non nulle d'atteindre un état absorbant (éventuellement en plusieurs étapes).

Derive in detail (mathematically) how to compute the absorbing probabilities (probability of being absorbed in an absorbing state).

Un état est dit **absorbant** s'il est impossible d'en sortir, c'est-à-dire si $p_{ii} = 1$. Les autres états sont appelés **transitoires**.

On peut réécrire la matrice de transition P sous forme canonique :

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$$

où :

- Q est la matrice de transition entre les états transitoires ;
- R est la matrice des probabilités de transition entre états transitoires et absorbants ;
- 0 est une matrice nulle ;
- I est la matrice identité correspondant aux états absorbants.

Les matrices Q et R sont dites **sub-stochastiques** (la somme des lignes est inférieure ou égale à 1).

Pour calculer P^t , on peut simplement se concentrer sur la puissance Q^t . Comme Q est sub-stochastique, alors $Q^t \rightarrow 0$ lorsque $t \rightarrow \infty$.

On définit alors la **matrice fondamentale** de la chaîne absorbante par :

$$\begin{aligned} N &= \sum_{t=0}^{\infty} Q^t \\ &= I + Q + Q^2 + Q^3 + \dots \\ &= (I - Q)^{-1} \quad \text{car c'est une suite géométrique} \end{aligned}$$

Les éléments n_{ij} de cette matrice, pour i, j correspondant à des états transitoires, s'interprètent comme :

$$\begin{aligned} n_{ij} &= e_i^T N e_j \\ &= e_i^T \left(\sum_{t=0}^{\infty} Q^t \right) e_j \\ &= \sum_{t=0}^{\infty} e_i^T Q^t e_j \\ &= \sum_{t=0}^{\infty} x_{j|i}(t) \quad \text{car, dans ce contexte, } Q^t = P^t \text{ puisque l'on considère uniquement les états transitoires} \\ &= \mathbb{E}[\text{nombre de visites en l'état } j \mid \text{départ en l'état } i] \end{aligned}$$

Cela signifie que n_{ij} est le nombre espéré de passages par l'état transitoire j lorsque l'on part de l'état i . Ainsi, le **nombre total de visites avant absorption** en partant de chaque état est donné par $n = Ne$ où e est un vecteur colonne de 1.

On peut maintenant calculer les **probabilités d'absorption**. La probabilité d'être absorbé par l'état absorbant j en partant de l'état transitoire i est donnée par :

$$b_{ij} = \sum_{t=0}^{\infty} \sum_{k=1, k \in \mathcal{T}_R}^{n_{tr}} x_{k|i}(t) r_{kj}$$

où :

- \mathcal{T}_R est l'ensemble des états transitaires ;
- $x_{k|i}(t)$ est la probabilité d'être en état k à l'instant t en partant de i ;
- r_{kj} est l'élément (k, j) de la matrice R .

Cette expression peut être simplifiée avec la matrice fondamentale N :

$$\begin{aligned} b_{ij} &= \sum_{t=0}^{\infty} \sum_{k=1, k \in \mathcal{T}_R}^{n_{tr}} x_{k|i}(t) r_{kj} \\ &= \sum_{k=1, k \in \mathcal{T}_R}^{n_{tr}} \left[\sum_{t=0}^{\infty} e_i^T Q^t e_k \right] r_{kj} \\ &= \sum_{k \in \mathcal{T}_R} n_{ik} r_{kj} \\ &= [NR]_{ij} \end{aligned}$$

Et donc, en forme matricielle, les probabilités d'absorption sont stockées dans la matrice $B = NR$ où chaque élément b_{ij} donne la probabilité d'être absorbé par l'état j en partant de l'état i .

Also discuss some game/finance application.

Note: Voir Chapitre 8 car cette question est communes à plusieurs questions.

Les chaînes de Markov trouvent des applications dans de nombreux domaines, notamment le filtrage collaboratif et le marketing.

Filtrage collaboratif: Un exemple d'application est l'algorithme *ItemRank*, utilisé pour recommander des articles ou produits à un utilisateur en se basant sur les transitions probables entre éléments. La navigation d'un utilisateur entre articles est modélisée comme un processus stochastique, dans lequel chaque item correspond à un état de la chaîne.

Cet exemple est défini en détail dans la Question 14.

Marketing: Dans un contexte de marketing relationnel, on peut segmenter les clients en différents clusters à l'aide du modèle RFM (Récence, Fréquence, Valeur Monétaire). Chaque cluster représente un état d'une chaîne de Markov. Le dernier cluster (souvent le n -ième) représente les clients perdus ; cet état est alors modélisé comme absorbant (on ne peut plus en sortir).

En observant les transitions mensuelles des clients entre clusters, on peut estimer les probabilités de transition. En associant un bénéfice moyen à chaque cluster, on peut alors calculer le **profit espéré généré par un client au cours de son cycle de vie**, également appelé *customer lifetime value* (CLV). Cette approche permet de mieux orienter les efforts de rétention client.

Cet exemple est défini en détail dans la Question 20.

Autres domaines d'application:

- **Reconnaissance vocale et traitement du langage** : les modèles de Markov (souvent cachés – HMM) permettent de représenter la structure probabiliste des séquences de sons ou de mots.
- **Modélisation du comportement utilisateur** : dans les jeux vidéo, ou sur des plateformes en ligne, pour anticiper les actions futures d'un joueur ou d'un internaute.
- **Finance** : modélisation de l'évolution de la solvabilité d'un emprunteur ou du passage entre différents états de crédit.

Question 21 - Markov Chain (Lifetime Value)

Describe and derive in detail (mathematically) how the « lifetime value » of a customer can be computed according to a Markov chain model (marketing application).

Describe and derive in detail (mathematically) how the « lifetime value » of a customer can be computed according to a Markov chain model (marketing application).

Supposons que nous ayons regroupé les clients d'une entreprise en n clusters (ou segments) à l'aide d'un critère tel que le modèle RFM (*Récence, Fréquence, Valeur Monétaire*).

Le modèle RFM est une méthode d'analyse comportementale des clients basée sur trois critères :

- la récence (date du dernier achat),
- la fréquence (nombre d'achats effectués sur une période donnée),
- et la valeur monétaire (montant total dépensé).

Ce modèle permet d'identifier les clients les plus engagés et les plus rentables pour orienter les actions marketing de manière ciblée.

Chaque cluster est modélisé comme un état d'une chaîne de Markov. Le dernier cluster, soit le n -ième, représente les clients perdus : cet état est absorbant, car une fois cet état atteint, le client quitte définitivement l'entreprise et ne génère plus de profit.

Estimation des probabilités de transition

Chaque mois, on observe les transitions des clients d'un cluster à un autre. On estime les probabilités de transition p_{ij} à partir des fréquences observées, ce qui permet de construire la matrice de transition P .

Profits et horizon infini

On suppose que chaque état i génère un profit moyen m_i par client. Ce profit peut être négatif (par exemple, en cas de coût élevé d'entretien client).

On considère également un facteur d'actualisation $0 < \gamma < 1$ pour prendre en compte la valeur temporelle de l'argent (les profits futurs ont moins de valeur que les profits immédiats).

Valeur à vie du client (CLV)

On note $x_i(t)$ la probabilité qu'un client soit dans l'état i au temps t . La valeur à vie du client, notée \bar{m} , correspond au profit actualisé total espéré généré par un client jusqu'à ce qu'il soit absorbé (perdu) :

$$\bar{m} = \sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^n x_i(t) m_i$$

On pose :

- $x(0)$: vecteur de distribution initiale des clients dans les états,
- m : vecteur des profits moyens par état, avec $m_n = 0$ (état absorbant),
- P : matrice de transition entre états.

L'évolution de la distribution de probabilité suit la dynamique : $x(t)^T = x(0)^T P^t$.

En injectant cela dans la formule, on obtient :

$$\bar{m} = \sum_{t=0}^{\infty} \gamma^t x(0)^T P^t m = x(0)^T \left(\sum_{t=0}^{\infty} \gamma^t P^t \right) m$$

On reconnaît ici une somme de type géométrique matricielle :

$$\sum_{t=0}^{\infty} \gamma^t P^t = (I - \gamma P)^{-1}$$

Ce qui nous donne la formule finale de la **Customer Lifetime Value (CLV)** :

$$\bar{m} = x(0)^T (I - \gamma P)^{-1} m$$

Interprétation

Cette formule permet de quantifier de manière rigoureuse le profit espéré d'un client sur un horizon infini, en tenant compte :

- des comportements de transition entre segments (fidélité, attrition, rétention),
- de la rentabilité moyenne par segment,
- du temps via l'actualisation (γ).

Elle permet ainsi aux entreprises d'estimer la rentabilité de leurs actions marketing, d'optimiser leurs campagnes de rétention et de cibler les bons segments de clientèle.

Question 22 - HMM & Likelihood

Explain the general principles and the structure of a hidden Markov model for speech recognition. Moreover, without going into the deep details, how can we solve the first important problem in HMMs, namely computing the likelihood of the observations (the probability of generating a sequence of observations). And finally, how do we recognize an uttered word among a finite dictionary of words.

Explain the general principles and the structure of a hidden Markov model for speech recognition.

Un **modèle de Markov caché** (HMM, pour *Hidden Markov Model*) est constitué d'états cachés (en vert), qui obéissent à la propriété de Markov — c'est-à-dire qu'un état donné dépend uniquement de l'état précédent — et d'observations (en violet), qui sont émises par ces états cachés.

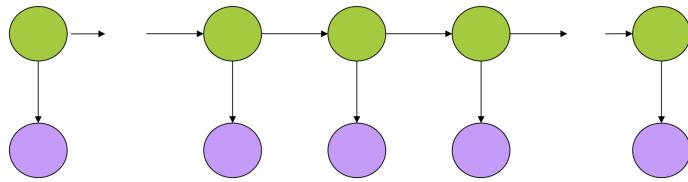


Figure 21.1: Illustration de la structure d'un HMM.

Les observations dépendent uniquement de l'état caché associé, à partir duquel elles sont générées.

On définit les quantités suivantes :

- $s(t) \in \{1, \dots, n\}$: variable aléatoire représentant l'état caché à l'instant t .
- $x(t) \in \{o_1, \dots, o_p\}$: variable aléatoire représentant l'observation à l'instant t , avec o_k une valeur discrète possible.
- $x = [x_1, \dots, x_T]^T$: séquence d'observations sur T instants.

Les paramètres du modèle sont :

- $\pi_i = \mathbb{P}(s_1 = i)$: probabilité initiale d'être dans l'état i .
- $P = (p_{ij})$ avec $p_{ij} = \mathbb{P}(s(t+1) = j \mid s(t) = i)$: matrice de transition entre états cachés.
- $B = (b_i(o_k))$ avec $b_i(o_k) = \mathbb{P}(x(t) = o_k \mid s(t) = i)$: probabilité d'émission de l'observation o_k depuis l'état i .

L'ensemble des paramètres est noté $\theta = \{\pi, P, B\}$.

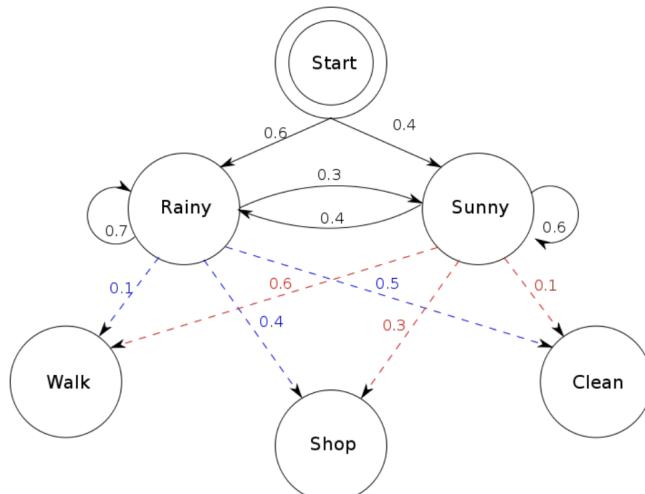


Figure 21.2: Illustration d'un exemple HMM.

Application : reconnaissance vocale

Dans le cas de la reconnaissance vocale, le signal audio capté représente les observations. Le but est d'identifier le mot correspondant à ce segment audio, autrement dit, retrouver la séquence d'états cachés (sons ou phonèmes) la plus probable à l'origine de ces observations.

Moreover, without going into the deep details, how can we solve the first important problem in HMMs, namely computing the likelihood of the observations (the probability of generating a sequence of observations).

Étant donné une séquence d'observations $x = [x_1, \dots, x_T]^T$ et un modèle θ , on souhaite calculer la vraisemblance $\mathbb{P}(x | \theta)$, c'est-à-dire la probabilité que le modèle génère cette séquence d'observations.

Sans optimisation, cela revient à sommer sur toutes les séquences d'états possibles :

$$\mathbb{P}(x | \theta) = \sum_{s_1, \dots, s_T} \pi_{s_1} b_{s_1}(x_1) \prod_{t=1}^{T-1} p_{s_t, s_{t+1}} b_{s_{t+1}}(x_{t+1})$$

Cependant, cette somme est coûteuse (exponentielle). On utilise donc les procédures **forward** et **backward**, qui permettent un calcul efficace via récurrence. **Procédure forward** On définit :

$$\alpha_i(t) = \mathbb{P}(x_1, \dots, x_t, s_t = i | \theta)$$

La récurrence s'écrit :

$$\begin{cases} \alpha_j(1) = \pi_j b_j(x_1) \\ \alpha_j(t+1) = (\sum_{i=1}^n p_{ij} \alpha_i(t)) b_j(x_{t+1}) \end{cases}$$

Ainsi, on obtient la vraisemblance comme :

$$\mathbb{P}(x | \theta) = \sum_{j=1}^n \alpha_j(T)$$

Procédure backward On définit :

$$\beta_i(t) = \mathbb{P}(x_{t+1}, \dots, x_T | s_t = i, \theta)$$

La récurrence est la suivante :

$$\begin{cases} \beta_i(T) = 1 \\ \beta_i(t) = \sum_{j=1}^n p_{ij} b_j(x_{t+1}) \beta_j(t+1) \end{cases}$$

On obtient au final :

$$\mathbb{P}(x | \theta) = \sum_{i=1}^n \pi_i \beta_i(1)$$

Combinaison des deux procédures Enfin, à tout instant t , on peut aussi combiner les deux approches :

$$\mathbb{P}(x | \theta) = \sum_{i=1}^n \alpha_i(t) \beta_i(t)$$

Cela permet également de calculer la probabilité marginale d'être dans un état donné au temps t (utilisée pour l'inférence).

Tous les développements détaillés figurent dans les slides du cours et ne sont pas repris ici, conformément à la consigne de ne pas entrer dans les aspects techniques. De manière générale, ces calculs reposent sur des règles classiques de manipulation des probabilités, en particulier à l'aide des quantités $\alpha_i(t) = \mathbb{P}(x_1, \dots, x_t, s_t = i | \theta)$ et $\beta_i(t) = \mathbb{P}(x_{t+1}, \dots, x_T | s_t = i, \theta)$, qui correspondent respectivement aux probabilités avant (forward) et arrière (backward) dans le cadre des modèles de Markov cachés.

And finally, how do we recognize an uttered word among a finite dictionary of words.

Étant donné un signal audio (c'est-à-dire une séquence d'observations), l'objectif est d'identifier **quel mot** du dictionnaire a le plus probablement été prononcé.

1. **Modélisation de chaque mot** : Chaque mot $w^{(k)}$ du dictionnaire (où $k = 1, \dots, K$) est associé à un modèle de Markov caché (HMM) $\theta^{(k)} = \{\pi^{(k)}, P^{(k)}, B^{(k)}\}$. Chaque modèle est appris à partir d'exemples de prononciation de ce mot.

2. **Observation d'un signal** : On extrait une séquence d'observations $x = [x_1, x_2, \dots, x_T]$ à partir du signal.
3. **Calcul des vraisemblances** : Pour chaque modèle $\theta^{(k)}$, on calcule la probabilité (vraisemblance) $\mathbb{P}(x | \theta^{(k)})$ en utilisant l'algorithme **forward**.
4. **Décision** : Le mot reconnu est celui dont le modèle maximise la vraisemblance :

$$\hat{k} = \arg \max_k \left\{ \mathbb{P}(x | \theta^{(k)}) \right\}$$

Le mot reconnu est alors $w^{(k)}$.

Le calcul exact de $\mathbb{P}(x | \theta^{(k)})$ nécessite de sommer sur tous les chemins possibles dans le modèle, ce qui peut être coûteux. De plus, pour une tâche de reconnaissance en temps réel et avec un grand dictionnaire, cette méthode devient rapidement impraticable. Cela revient à faire une sorte de KNN avec $k = 1$ avec une "distance" qui est la vraisemblance.

Algorithme Viterbi (algorithme DP)

1. On modélise chaque mot $w^{(k)}$ avec un HMM $\theta^{(k)}$.
2. Pour une séquence observée $x = [x_1, \dots, x_T]$, on exécute l'algorithme de Viterbi pour chaque modèle :
 - On calcule $\delta_j(t)$, la probabilité du chemin le plus probable finissant en l'état j à l'instant t .
 - On applique :

$$\begin{aligned}\delta_j(1) &= \pi_j b_j(x_1) \\ \delta_j(t+1) &= \max_i \{ \delta_i(t) \cdot p_{ij} \cdot b_j(x_{t+1}) \} \\ \psi_j(t+1) &= \arg \max_i \{ \delta_i(t) \cdot p_{ij} \cdot b_j(x_{t+1}) \}\end{aligned}$$

En pratique, on utilise le logarithme de cette quantité pour éviter les problèmes d'underflow sur ordinateur.

3. À la fin, on récupère le score maximum $\max_j \delta_j(T)$ pour chaque modèle.
4. On choisit le mot dont le **chemin le plus probable** donne la probabilité la plus élevée :

$$\hat{k} = \arg \max_k \left(\max_j \delta_j^{(k)}(T) \right)$$

Question 23 - Fair Prediction in SC

Explain in detail how we can introduce the notion of “fair prediction” in supervised classification, as well as how to mitigate discrimination when using a logistic regression model.

Explain in detail how we can introduce the notion of “fair prediction” in supervised classification.

La plupart des techniques s'appuient sur la notion de groupes sensibles ou protégés, dont la propriété ne doit pas être prise en compte dans la prédiction et qui doivent être protégés contre la discrimination. Ce besoin est particulièrement important dans les contextes où les décisions algorithmiques affectent des aspects critiques de la vie (emploi, prêts, admissions universitaires, etc.).

Cependant, retirer simplement la variable protégée du modèle n'est souvent pas suffisant, car certaines variables explicatives (features) peuvent être fortement corrélées avec la variable protégée, agissant comme des proxy implicites et perpétuant la discrimination.

Dans un problème de classification supervisée, nous considérons :

- X : la matrice des données,
- Z : la variable protégée (souvent binaire, ex : genre, ethnie),
- Y : la variable cible (l'outcome réel),
- \hat{Y} : la prédiction probabiliste du modèle,
- \hat{Y}_d : la décision binaire finale après seuillage.

Plusieurs cadres formels permettent de définir l'équité :

A) Équité au niveau des groupes (Group fairness)

- Parité statistique (ou démographique) :

$$\mathbb{P}(\hat{Y}_d = 1 \mid Z = 1) = \mathbb{P}(\hat{Y}_d = 1 \mid Z = 0)$$

Cette mesure exige que la probabilité de recevoir une prédiction positive soit égale entre les groupes protégés et non protégés.

- Parité attendue sur les scores :

$$\mathbb{E}[\hat{Y} \mid Z = 1] = \mathbb{E}[\hat{Y} \mid Z = 0]$$

Cette variante s'applique aux scores de probabilité plutôt qu'aux décisions binaires.

- Relaxation par seuil ϵ :

$$\left| \mathbb{P}(\hat{Y}_d = 1 \mid Z = 1) - \mathbb{P}(\hat{Y}_d = 1 \mid Z = 0) \right| \leq \epsilon$$

Permet une légère différence entre les groupes, reconnaissant les contraintes pratiques.

- Parité démographique conditionnelle :

$$\left| \mathbb{P}(\hat{Y}_d = 1 \mid X, Z = 1) - \mathbb{P}(\hat{Y}_d = 1 \mid X, Z = 0) \right| \leq \epsilon$$

Importante pour éviter le paradoxe de Simpson, qui survient lorsqu'une variable X est associée à la fois à Y et Z , créant une apparence trompeuse de discrimination.

- Égalité des chances :

$$\mathbb{P}(\hat{Y}_d = 1 \mid Y = 1, Z = 1) = \mathbb{P}(\hat{Y}_d = 1 \mid Y = 1, Z = 0)$$

Exige que les taux de vrais positifs soient égaux entre les groupes.

- Odds égalisés : combinaison de l'égalité des chances et de l'égalité des taux de faux positifs :

$$\mathbb{P}(\hat{Y}_d = 1 \mid Y = 1, Z = 1) = \mathbb{P}(\hat{Y}_d = 1 \mid Y = 1, Z = 0) \quad \text{et} \quad \mathbb{P}(\hat{Y}_d = 1 \mid Y = 0, Z = 1) = \mathbb{P}(\hat{Y}_d = 1 \mid Y = 0, Z = 0)$$

B) Équité individuelle (Individual fairness)

Des individus similaires devraient recevoir des traitements similaires :

$$\text{dist}_Y(\hat{y}_i, \hat{y}_j) \leq \epsilon \times \text{dist}_X(x_i, x_j)$$

Cette formulation exige que la différence entre les prédictions pour deux individus soit proportionnelle à leur différence dans l'espace des caractéristiques. Le défi pratique réside dans le choix approprié des fonctions de distance et du facteur d'échelle ϵ .

C) Équité basée sur la causalité

Cette approche cherche à déterminer si la variable protégée a un effet causal sur la décision, ce qui est généralement difficile à vérifier pour des données observationnelles.

Il est important de noter que certaines de ces définitions d'équité peuvent être mathématiquement incompatibles entre elles, forçant souvent un compromis lors de l'implémentation.

Explain in detail how to mitigate discrimination when using a logistic regression model.

La correction de l'équité conduit généralement à un compromis entre équité et performance prédictive du modèle. Pour un jeu de données donné, la covariance entre la variable protégée Z et la prédiction \hat{Y} est proportionnelle à la parité démographique molle (soft demographic parity) :

$$\text{cov}(Z, \hat{Y}) \propto \mathbb{E}[\hat{Y} | Z = 1] - \mathbb{E}[\hat{Y} | Z = 0]$$

Trois approches principales existent pour mitiger la discrimination :

A) Algorithmes de pré-traitement (pre-processing)

Ces méthodes visent à améliorer l'équité de la matrice de données avant l'apprentissage du modèle. Une technique courante consiste à :

1. Régresser chaque variable X_k en fonction des variables protégées Z , afin de voir quelle partie de X_k est prévisible à partir de Z :

$$x_{ik} = w_1 z_{i1} + w_2 z_{i2} + \dots + e_{ik}$$

où $e_{ik} = x_{ik} - \hat{x}_{ik}$ est le résidu, c'est-à-dire la partie de X_k qui ne dépend pas de Z . \hat{x}_{ik} est la valeur prédictée de la variable x_{ik} par la régression sur les variables protégées Z .

2. Ces résidus e_{ik} sont par construction non corrélés linéairement avec Z .
3. Remplacer les variables originales X_k par leurs résidus e_{ik} dans la matrice de données.

Cette approche purifie les variables explicatives en retirant leur association linéaire avec les variables protégées, mais n'élimine pas nécessairement toutes les formes de dépendance.

B) Algorithmes d'apprentissage (in-training)

L'équité est intégrée directement dans la phase d'apprentissage du modèle, soit :

- Par **régularisation** : ajout d'un terme pénalisant la dépendance entre \hat{Y} et Z dans la fonction objectif.

Dans le contexte d'un **modèle de régression logistique** avec **l'information mutuelle**, plus le terme de régularisation est petit, plus l'indépendance entre la prédiction du modèle et la variable sensible est grande.

C'est très intéressant, car cela encourage une véritable indépendance vis-à-vis des variables sensibles.

En revanche, cela rend le problème d'optimisation beaucoup plus difficile à résoudre.

- Par **contrainte** : ajout d'une contrainte d'équité dans le problème d'optimisation.

Ils contraignent la covariance entre la variable sensible et le score linéaire prédit fourni par la régression logistique à être faible, c'est-à-dire en dessous d'un certain seuil ϵ .

Le score linéaire prédit est $w^T x$ et est proportionnel à la distance de x à l'hyperplan de séparation. Ce score est ensuite passé dans une fonction sigmoïde non linéaire afin d'obtenir les prédictions du modèle.

Le problème d'optimisation constraint (pour un problème binaire) est :

$$\min_w \left\{ - \sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) \right\}$$

sous la contrainte

$$\frac{1}{n} |z^T H X w| \leq \epsilon$$

où la fonction objective minimise la fonction de perte log-vraisemblance négative (souvent appelée entropie croisée).

Rappelons que les sorties (scores prédictifs) du modèle de régression logistique sont $w^T x$.

Un inconvénient de cette méthode est qu'elle n'impose pas directement l'équité (fairness) sur les scores prédictifs. Cependant, ce modèle peut être étendu aux machines à vecteurs de support (SVM).

- **Régression logistique à entropie maximale** : le modèle maximise la somme des entropies des appartenances aux classes sous des contraintes de co-moments, c'est-à-dire en imposant que les co-moments entre les variables explicatives et les appartenances aux classes soient préservés :

$$\max_{\hat{y}_{ik}} \left\{ - \sum_{i,k} \hat{y}_{ik} \log \hat{y}_{ik} \right\}$$

sous les contraintes:

$$\begin{cases} \hat{\mathbf{y}}_k^T \mathbf{e} = n & \forall k \\ \frac{1}{n} \mathbf{X}^T \hat{\mathbf{y}} = \frac{1}{n} \mathbf{X}^T \mathbf{y} \\ \left| \frac{1}{n-1} \sum_i \hat{y}_{ik} \tilde{z}_i \right| \leq \epsilon \end{cases}$$

Le principe du maximum d'entropie repose sur l'idée que l'on doit uniquement utiliser les observations disponibles, sans ajouter d'informations superflues pour la construction du modèle. Ainsi, le modèle doit être *maximamente entropique*, c'est-à-dire aussi parcimonieux que possible en termes d'information.

Plus précisément, le problème convexe d'optimisation consiste à contraindre les co-moments observés et prédictifs à être égaux, assurant ainsi la préservation des co-moments entre les variables explicatives et les appartenances aux classes.

Par ailleurs, il est facile d'ajouter des contraintes sur les valeurs prédictives (scores) du modèle.

Avantages: Cette formulation présente plusieurs avantages :

- Elle facilite l'ajout de contraintes directement sur les valeurs prédictives, c'est-à-dire les appartenances aux classes.
- La contrainte de parité démographique peut être appliquée directement sur ces appartenances.

Limites: Lorsque le paramètre ϵ , qui contrôle la force des contraintes, tend vers zéro, les contraintes peuvent devenir incompatibles ou incohérentes. Dans ce cas, on commence par résoudre un problème de programmation linéaire simple qui recherche la covariance entre la variable sensible et la prédiction la plus proche de zéro. Cela permet d'obtenir la meilleure covariance réalisable, limitant ainsi la discrimination.

Problème d'optimisation par programmation linéaire: On suppose que la covariance entre la variable sensible et la valeur prédictive est négative, ce qui indique une discrimination. Si cette covariance n'est pas négative, aucun traitement n'est nécessaire.

C) Algorithmes de post-traitement (post-processing)

Ces techniques modifient les décisions finales pour atteindre l'équité souhaitée. L'avantage est qu'elles peuvent s'appliquer à n'importe quel modèle de classification déjà entraîné.

Une approche typique utilise l'échange itératif :

1. Entraîner le modèle de classification normalement.
2. Appliquer des échanges :
 - échanger les individus protégés rejetés les plus proches de la frontière de décision vers acceptés,

- échanger les individus non protégés acceptés les plus proches de la frontière vers rejetés,
- continuer jusqu'à ce que

$$\left| \mathbb{P}(\hat{Y}_d = 1 \mid Z = 1) - \mathbb{P}(\hat{Y}_d = 1 \mid Z = 0) \right| \leq \epsilon.$$

Cette méthode est simple mais peut être sous-optimale car découpée de l'apprentissage du modèle.

