Cours IFT6266,
Apprentissage Non-Supervisé de Variétés

L'Apprentissage Non-Supervisé

- L'apprentissage non-supervisé vise à caractériser la distribution des données, et les relations entre les variables, sans discriminer entre les variables observées et les variables à prédire.
- Les formes principales d'apprentissage non-supervisé sont les suivantes:
  - L'estimation de fonction de densité ou de fonction de probabilité. C'est la forme la plus générale d'apprentissage non-supervisé. On a un critère clair, la log-vraisemblance (mais certains remettent cela en question). On apprend explicitement la fonction $p(x)$.
  - La découverte de classes naturelles, ou *clustering* (e.g., l'algorithme K-moyennes), qui cherche à découvrir les **modes** principaux de la distribution, les "prototypes", les catégories principales, etc... Cela donne une forme de réduction de dimensionalité qui associe un petit entier à chaque exemple.
  - L'apprentissage de variétés de faible dimension, c'est à dire de surfaces (planes ou non-linéaires) près desquelles se retrouvent la majorité des données, en haute dimension. On obtient ainsi une représentation de faible dimension des données, ce qui peut être une étape importante pour visualiser les données et/ou comme prétraitement avant l'apprentissage supervisé. Très utile en finance pour réduire la dimensionalité et éviter l'overfitting.

La classification non-supervisée

Il s'agit de découvrir les modes principaux (classes) de la distribution.

Approche classique: une classe est caractérisée par son centre (un prototype).

Beaucoup de critères et d'algorithmes ont été proposés.

Critère le plus commun: erreur de reconstruction (reconstruire l'entrée à partir de sa classe coûte la distance entre l'entrée et le prototype central de la classe):

$$C = \sum_{i=1}^{n} \min_j ||x_i - \mu_j||^2$$

Algorithme des k-moyennes (*K-means*)
Données = $D = \{x_1, \ldots, x_n\}$
hyper-paramètre = k = nb classes
paramètres appris = μ_i, $i = 1$ à k = centres des classes.

1. Initialiser les μ en choisissant $\mu_i = x_j$ au hasard dans D.
2. Itérer jusqu'à ce que l'erreur de reconstruction ne baisse plus.
   (a) Pour $i = 1$ à n, prendre l'assignation $s_i = \text{argmin}_j ||x_i - \mu_j||^2$
   (b) L'erreur de reconstruction est $C = \sum_i ||x_i - \mu_{s_i}||^2$.
   (c) Pour $j = 1$ à k, $\mu_j = \frac{\sum_i x_i 1_{s_i=j}}{\sum_i 1_{s_i=j}}$ = la moyenne des exemples associés au prototype j

L'algorithme converge vite mais est sensible aux minima locaux.

Version plus robuste: **Hierarchical Clustering**

1. Trouver d'abord une classification en 2 classes.
2. Partitionner D en deux sous-ensembles D_1 et D_2 associés à ces deux classes.
3. Récursivement partitionner chacune...
4. Critère d'arrêt nécessaire: quand le gain apporté par la division d'une classe en deux est inférieur à un seuil.

Comment choisir le nombre de classes k?
k est un hyper-paramètre mais on ne peut pas utiliser l'erreur de reconstruction pour le choisir (sur un ensemble de validation), car plus k est grand est plus l'erreur de reconstruction sera petite, même sur l'ensemble de validation.
Une solution simple est de construire le mélange de gaussiennes correspondant au choix des μ_i et utiliser ensuite la log-vraisemblance sur l'ensemble de validation. Pour construire le mélange de gaussiennes correspondant, il faut choisir les variances (possiblement les covariances) σ_i^2 et les poids $P(i)$ associés à chaque gaussienne dans le mélange. Pour ce faire il suffit de considérer la variance (possiblement la covariance) à l'intérieur du sous-ensemble des exemples assignés, et de prendre $P(i)$ = proportion des exemples assignés à la gaussienne i. Optionnellement on peut partir de ce point-là et appliquer itérativement l'algorithme EM pour améliorer la vraisemblance.
Discussion à venir: clustering vs multi-clustering, i.e. versions *distribuées* du clustering (qui est un algorithme "local").

L'Analyse en Composantes Principales (PCA)

- On cherche une variété affine qui passe le plus près possible des données $x_1, \ldots, x_n$.
- On l'obtient avec l'algorithme suivant:
  - Soustraire la moyenne $\bar{x} = \frac{1}{n} \sum_i x_i$ des données: $\tilde{x}_i = x_i - \bar{x}$.
  - Calculer la matrice de covariance empirique des données: $C = \frac{1}{n-1} \sum_i \tilde{x}_i \tilde{x}_i'$.
  - Calculer les k vecteurs propres principaux v_j de $C = \sum_{j=1}^n \lambda_j v_j v_j'$.

– La i-ème coordonnée réduite de x est simplement $x.v_i$ (ou bien, pour obtenir des coordonnées de variance unitaire, $\frac{x.v_i}{\sqrt{\lambda_i}}$).

Les points sur la variété affine satisfont l'équation $\hat{x} = \sum_{i=1}^{k} \alpha_i v_i$ où α_i ($i = 1$ à k) représentent un nouveau système de coordonnées. Par l'orthogonalité des v_i, on peut obtenir ces coordonnées simplement en projetant un exemple x sur l'affine, donc sur les k vecteurs propres principaux: $\alpha_i = x.v_i$. On peut vérifier aisément que $(x - \hat{x})$ est orthogonal à $\hat{x}$, i.e.

$$\begin{aligned}(x-\hat{x}).\hat{x} &= (x - \sum_i v_i v_i' x)'(\sum_j v_j v_j' x) \\ &= x'(\sum_j v_j v_j')x - \sum_{i,j} x' v_i v_i' v_j v_j' x \\ &= x'(\sum_j v_j v_j')x - x'(\sum_j v_j v_j')x = 0\end{aligned}$$

en utilisant l'orthonormalité des vecteurs propres, $v_i.v_j = 1_{i=j}$.

- Notons que les coordonnées $\hat{x}_i$ de $\hat{x}$ tel que défini ci-haut ont comme variance λ_i. Souvent on veut une représentation des données dont la distribution est approximativement normale(0,1) dans toutes les directions ("**sphériser**" la distribution). Dans ce cas on utilise les **projections normalisées**, dont les coordonnées sont $\frac{x.v_i}{\sqrt{\lambda_i}}$.

Voir feuillet `pca.pdf` sur la page du cours.

Le *Multi-Dimensional Scaling* (MDS)

- Parfois on a pas les coordonnées des exemples, mais seulement les distances (ou autre mesure de similarité) entre chaque paire d'exemples. Le MDS classique trouve une représentation des exemples qui correspond exactement à la PCA, mais en partant de ces distances D_{ij}.
- L'algorithme est le suivant:
  - Moyennes par rangées: $\mu_i = \frac{1}{n}\sum_j D_{ij}$.
  - Double centrage (distance vers produit scalaire): $P_{ij} = -\frac{1}{2}(D_{ij} - \mu_i - \mu_j + \frac{1}{n}\sum_i \mu_i)$
  - Calcul des vecteurs propres v_j et valeurs propres λ_j principales de la matrice P (avec λ_j^2 plus grand).
  - La i-ème coordonnée réduite de l'example j est $\sqrt{\lambda_i} v_{ji}$.
- Notez bien: seuls les exemples d'apprentissage recoivent une coordonnée réduite.
- On peut généraliser le MDS pour permettre des variétés *non-linéaires* mais on obtient une fonction de coût non-convexe qui peut être difficile à optimiser.

Spectral Clustering

- C'est un algorithme de clustering qui consiste en deux étapes: d'abord la transformation des données en coordonnées réduites (correspondant à des variétés non-linéaires), suivi d'une étape classique de clustering (comme l'algorithme k-moyennes).
- Algorithme:
  - Appliquer un noyau $K(x_i, x_j)$ à chaque paire d'exemples (x_i, x_j).
  - Normaliser le noyau: $K_{ij} = \frac{K(x_i,x_j)}{\sqrt{\mu_i \mu_j}}$ où $\mu_i = \frac{1}{n}\sum_{i=1}^{n} K(x_i, x_j)$ est la moyenne par rangée.
  - Calculer les vecteurs propres principaux v_j de la matrice K.
  - Obtenir les coordonnées de norme 1 pour chaque exemple i: $(v_{1i}, v_{2i}, \ldots, v_{ki})/\sqrt{\sum_j v_{ji}^2}$.
  - Appliquer un algorithme de clustering classique (k-moyennes) sur les exemples dans ce nouveau système de coordonnées.

Local Linear Embedding **(LLE)**

- C'est une méthode pour découvrir une variété non-linéaire basée sur l'approximation dans un espace de faible dimension des relations géométriques locales dans chaque régions délimitée par les k plus proches voisins d'un exemple.
- Algorithme:
  - Trouver les m plus proches voisins x_j de chaque example x_i.
  - Trouver pour chaque exemple i les poids w_{ij} sur les voisins j qui minimisent l'erreur de régression $(x_i - \sum_j w_{ij} x_j)^2$, avec la contrainte $\sum_j w_{ij} = 1$. On prend $w_{ij} = 0$ si j n'est pas un voisin de i.
  - Transformer la matrice sparse de poids W en $M = (I-W)'(I-W)$ symmétrique.
  - Calculer les k vecteurs propres v_j de **plus petite valeur propre** (excluant la plus petite, qui est 0), ce qui donne les coordonnées réduites des exemples d'apprentissage, $(v_{1i}, v_{2i}, \ldots, v_{ki})$.
- Notez que les vecteurs propres sont les coordonnées réduites y_i pour chaque exemple x_i qui minimisent l'erreur

$$\sum_i ||y_i - \sum_j w_{ij} y_j||^2$$

sous la contrainte que $\sum_i y_{ij}^2 = 1$ (normalisation des coordonnées, sinon la solution est $y_i = 0$) et $\sum_i y_{ij} y_{ik} = 1_{j=k}$ (sinon il y a une infinité de solutions correspondant à des rotations des coordonnées). On cherche donc à reproduire la structure géométrique locale, mais avec un système de coordonnées de faible dimension.

ISOMAP

- Comme LLE, cet algorithme se base sur les relations linéaires locales entre voisins pour capturer la structure de la variété, mais il a aussi une composante "globale", en essayant de preserver les distances **le long de la variété**. Pour cela on essaie d'approximer la **distance géodésique** sur la variété par la distance minimale dans un graphe dont les noeuds sont les exemples et les arcs seulement entre voisins sont associées aux distances locales.
- L'algorithme est le suivant:
  - Calculer les m plus proches voisins de chaque exemple, avec les distances $d(x_i, x_j)$ correspondantes, pour peupler le graphe.
  - Calculer la longueur $D(x_i, x_j)$ du chemin le plus court dans le graphe entre chaque paire d'exemples: $D(x_i, x_j) = \min_p \sum_k d(p_k, p_{k+1})$ où p est un chemin $(p_1, p_2, ..p_l)$ entre x_i et x_j dans le graphe ($p_1 = x_i$, $p_l = x_j$).
  - Appliquer l'algorithme MDS sur la matrice des distances géodésiques, $D_{ij} = D(x_i, x_j)$, ce qui donne les coordonnées réduites pour les exemples d'apprentissage.

Kernel PCA

- On peut généraliser l'algorithme de la PCA avec le "truc du noyau".
- La matrice de covariance C dans l'espace des $\phi(x)$ (qui peut maintenant être de dimension infinie) est $C = \frac{1}{n} \sum_{i=1}^{n} (\phi(x_i) - \mu)(\phi(x_i) - \mu)'$ avec $\mu = \frac{1}{n} \sum_i \phi(x_i)$. Les vecteurs propres u_j de cette matrice sont liés aux vecteur propres v_j de la matrice de Gram K: $K_{ij} = K(x_i, x_j) - \bar{K}_i - \bar{K}_j + \bar{\bar{K}}$ et $\bar{K}_i = \frac{1}{n} \sum_j K(x_i, x_j)$, $\bar{\bar{K}} = \frac{1}{n} \sum_i \bar{K}_i$.
- On obtient la projection de $\phi(x)$ sur le j-eme vecteur principal de C (donc la j-ème coordonnée d'un exemple quelquonque x) avec

$$u_j.\phi(x) = \frac{1}{\lambda_j} \sum_i v_{ji} K(x, x_i)$$

où λ_j est la j-ème valeur propre de la matrice de Gram K et v_{ji} le i-ème élément du j-ème vecteur propre.

REFERENCES: pour les articles plus détaillés voir la page du cours pour le cours d'aujourd'hui, ainsi que les références des articles suivants, disponibles sur la page du LISA (`http://www.iro.umontreal.ca/~lisa`):

- Y. Bengio, J-F. Paiement, and P. Vincent. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. Technical Report 1238, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.

- Y. Bengio, P. Vincent, J-F. Paiement, O. Delalleau, M. Ouimet, and N. Le Roux. Spectral Clustering and Kernel PCA are Learning Eigenfunctions. Technical Report 1239, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.