PCA

On se donne une matrice $\mathbf{X} = (x_{i,j})$ avec $1 \leq i \leq n$ le nombre d'observations et $1 \leq j \leq p$ le nombre de variables.
Matrice de pondération des observations : D_w une matrice diagonale telle que $tr(D_w) = 1$. Généralement, chaque observation possède le même poids, on a donc $D_w = \frac{1}{n} I_n$.
Il est nécessaire de centrer $\mathbf{X}$: $\mathbf{Y} = (y_{i,j}) = (x_{i,j} - \overline{x_j})$ avec $\overline{x_j} = \sum_{i=1}^{n} w_i x_{i,j}$.
Matrice des covariances empiriques : $\mathbf{V} = (s_{i,j})$ avec $s_{i,j} = \sum_{k=1}^{n} w_k (x_{k,i} - \overline{x_i})(x_{k,j} - \overline{x_j}) = \sum_{k=1}^{n} w_k y_{k,i} y_{k,j}$.
Matrice des données réduites et standardisées : $\mathbf{Z} = (z_{i,j})$ avec $(z_{i,j}) = \frac{y_{i,j}}{s_j}$.
Matrice des coefficients de correlation : $\mathbf{R} = (r_{i,j})$ avec $(r_{i,j}) = \frac{s_{i,j}}{s_i s_j}$.
Remarque : $\mathbf{V}$ et $\mathbf{R}$ sont des matrices carrées de dimension p symétriques et semi-définies positives. Il y a deux types d'APC. L'APC canonique qui utilise $\mathbf{Y}$ et qui effectue son analyse sur $\mathbf{V}$, elle est utilisée généralement quand les variables sont dans la même unitée. L'APC standardisé qui utilise $\mathbf{Z}$ et qui effectue son analyse sur $\mathbf{V} = \mathbf{R}$.
Produit scalaire sur $\mathbb{R}^p$: on se donne une matrice $\mathbf{M}$ symétrique et définie positive. Alors $< u, v >= u^T M v$.
Pour un nuage d'observations ($\mathbb{R}^p$), on prend $\mathbf{M} = I_p$. Pour un nuage de variables ($\mathbb{R}^n$), on prends $\mathbf{M} = D_w$. C'est ce dernier cas qui nous interresse. On a alors : $< y_j, y_k >= s_{j,k}$, $||y_j||^2 = s_j^2$, $d^2(y_j, y_k) = var(y_j - y_k)$ et $cos(y_j, y_k) = r_{j,k}$.
Inertie totale du nuage d'observation : $\mathrm{I}(\mathbf{X}) = \sum_{i=1}^{n} w_i d^2(x_i, g) = \sum_{i=1}^{n} w_i ||x_i - g||^2 = \sum_{i=1}^{n} w_i ||y_i||^2 = \mathrm{I}(\mathbf{Y})$.
Remarques : l'inertie correspond à la variance au cas multivarié, $\mathbf{I(Z)} \neq \mathbf{I(Y)}$ et $\mathbf{I} = \mathrm{tr}(\mathbf{V})$.
Inertie expliquée par un sous espace vectoriel F (de dimension $q < p$) de $\mathbb{R}^p$: $I_F(\mathbf{Y}) = \sum_{i=1}^{n} w_i ||P(y_i)||^2$ avec P la projection orthogonale sur le sous espace.
Cas particulier en dimension 1 : si $F = vect(u)$ avec $||u|| = 1$, on a $I_F(\mathbf{Y}) = u^T V u$.
On construit par récurrence E_q notre sous espace vectoriel : $E_q = \Delta u_1 \oplus \Delta u_2 \oplus ... \oplus \Delta u_q$, avec u_i le vecteur normalisé associé au i-ème plus grand vecteur propre de $\mathbf{V}$ λ_i de la manière suivante : $I_{\Delta u_i}(Y) = \lambda_i$.
Nombre d'axes à retenir :
Critère d'inertie : $\frac{I_{E_q}}{I} = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq \alpha$ avec en général $\alpha = 0.8$.
Règle de Kaiser : $\lambda_q \geq \frac{I}{p}$ et $\lambda_{q+1} < \frac{I}{p}$.
Recherche d'un coude en faisant un graphique des valeurs propres dans l'ordre décroissant.
On note $c_j = Y u_j$ la projection de $\mathbf{Y}$ sur u_j. On nomme ce vecteur une composante principale. Sa moyenne est nulle et sa variance et λ_j.

Clustering

On a la même matrice $\mathbf{X}$ et on essaie de partitionner les observations $\mathcal{X} = \{\mathbf{x_1}, \cdots, \mathbf{x_i}, \cdots, \mathbf{x_n}\}$.
Distance de Minkosky : distance en utilisant la norme L_q. Distance Euclidienne quand $q = 2$ et distance de Manhattan quand $q = 1$. Distance de Chebyshev en utilisant la norme infinie. En utilisant la norme définie par le produit scalaire, on a la distance de Mahalanobis en prenant $M = V^{-1}$.
Pour les distances discrètes, la distance de Hamming est le nombre de différence, la distance de Levenshtein est le nombre d'ajout, de suppression et de remplacement. Enfin, en notant $a_{i,j}$ le nombre d'éléments en communs entre x_i et x_j, $b_{i,j}$ ceux qui apparraissent dans x_i mais pas dans x_j, ... On a le coefficient de simple matching $\frac{a_{i,j}+d_{i,j}}{p}$ et l'indice de Jaccard $\frac{a_{i,j}}{a_{i,j}+b_{i,j}+c_{i,j}}$.
Méthodes combinatoires
On cherche une partition de $\mathcal{X}$ en K clusters en minimisant une fonction de coût définie à partir d'un critère de similarité/dissimilarité. On notera g_k le centroïde du cluster C_k.
On a : $I = \sum_{i=1}^{n} ||x_i - g||^2 = \sum_{k=1}^{K} \sum_{x_i \in C_k} d^2(x_i, g_k) + \sum_{k=1}^{K} \#C_k d^2(g_k, g) = I_w + I_B$ avec I_w l'inertie intra-classes qu'on cherche à minimiser et I_B l'inertie inter-classes qu'on cherche à maximiser.
Algorithme des K moyennes : convergence garantie mais seulement vers un min local, faible complexité, facile à interpréter, sensible à l'initialisation, adapté aux clusters de formes convexes et de tailles équilibrées. Par contre, il est sensible aux valeurs aberrantes, ne trouve que des clusters convexes, et on doit fixer K en avance.
Clustering hierarchique agglomératif (AHC) : bottom up (ce qui est utilisé en général) ou top down. Il faut définir la distance inter classe pour l'utiliser. **Simple linkage** : distance entre les deux points les plus proches. Tendance à produire des clusters larges, sensible au bruit et aux valeurs aberrantes. **Complete linkage** : distance entre les deux points les plus éloignés. Tendance à produire des clusters compacts, sensible au bruit et aux valeurs aberrantes. **Average linkage** : moyenne des distances entre les points de C_a et de C_b. Tendance à construire des clusters plus homogènes, moins sensible aux valeurs aberrantes. **Centroïde linkage** : $d(C_a, C_b) = d(g_a, g_b)$. Moins sensible aux valeurs aberrantes.
Le nombre K de clusters n'est pas requis à l'avance, il suffit de faire un dendogramme. L'algorithme est totalement déterministe. La complexité est élevée.
Pour trouver K, on peut tracer I_w en fonction de K et trouver le coude. On peut aussi chercher le score silhouette le plus élevé : $s(x_i) = \frac{b(x_i)-a(x_i)}{max(a(x_i),b(x_i))}$ avec $a(x_i)$ la distance moyenne du point à son groupe et $b(x_i)$ la distance moyenne du point au groupe voisin.
Méthode basée sur la densité On veut trouver des clusters de formes complexes et qu'ils ne soient pas influencés par les valeurs aberrantes. On pose ϵ et n_{min}. On cherche les points centraux : ceux dont le nombre de points dans leur ϵ-voisinage est $\geq n_{min}$. tous les points dans ces ϵ-voisinages appartiennent aux clusters définis par les points centraux. On relie les clusters dont l'intersection n'est pas nulle. En général, on prend $n_{min} = 2p$. Pour trouver ϵ, on regarde pour tous les points la distance avec leur n_{min} i-ème voisin, on trace dans l'ordre décroissant et on prend le coude.

Classification

Classification discriminante : on cherche une ou plusieurs frontières, $P(y|x)$.
Classification générative : on cherche les clusters, $P(x|y)$.
Techniques d'encodage : one hot encoding, label encoding et ordinal encoding (on préserve un ordre).
Méthode des ensembles
Bagging (Bootsrap Aggregating) : on entraine k modèles sur une partition du jeu d'entraîement de cardinal k. Ensuite, on utilise ces modèles pour déterminer la classe finale d'une observation (on utilise souvent le vote majoritaire).
Boosting : à chaque itération, on entraîne un modèle en augmentant le poids des erreurs du modèle précédent.
Stacking : on utilise un méta-modèle qui prend en entré la sortie de plusieurs modèles simples et qui détermine la classe finale.
Interprétabilité
SHAP (SHapley Additive exPlanation) : attribue une valeur d'importance à chaque caractéristique d'entrée, indiquant son impact sur la prédiction finale.
LIME (Local Interpretabile Model-agnostic Explanation) : approxime localement le comportement d'un modèle complexe par un modèle interpretable.
Optimisation des hyper-paramètres
Grid search : brut force
Random search : on regarde aléatoirement les combinaisons d'hyper-paramètres et on choisis la meilleure.
Bayesian optimisation : construit un modèle probabiliste pour trouver la meilleure combinaison d'hyper-paramètres.
Modèles non linéaires Arbre de décision : on choisis la variable v qu'on veut utiliser en label. Pour chaque variable v_i, on la décompose en ses sous ensembles se_j. On isole ces sous ensembles et on calcul l'indice de Gini pour les sous ensembles associés de v : $1 - \sum p_k^2$. On pondère ces indices par la répartition des sous ensembles se_j. On choisis le v_i avec le score le plus faible.
t-SNE : modèle de réduction de dimension utilisé pour la visualisation.
Modèle linéaire
Régression logistique : donne la probabilité d'appartenir à une certaine classe. La fonction d'activation est la fonction sigmoïde : $\sigma(z) = \frac{1}{1+e^{-z}}$ avec $z = \beta_0 + \sum \beta_i x_i$. On doit maximiser la fonction de vraisemblance :
$L(\beta) = \sum((y_i log(\sigma(z_i)) + (1 - y_i) log(1 - \sigma(z_i)))$.
Perceptron : $w_i \leftarrow w_i + \alpha(y - \hat{y})x_i$.
Large Margin Classifier : on cherche à maximiser la marge entre les classes. Equation de l'hyperplan séparateur : $w^T x + b = 0$. On minimise $||w||$ sous contraintes $y_i(w^T x_i + b) \geq 1$ pour tous $i \in [\![1\ ;\ n]\!]$.
SVM : Large Margin Classifer + kernel trick pour creer des frontières non linéaires. Possibilité de classification multiclasse.

Régression

Modèle linéaire simple
$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, avec ϵ_i $N(0, \sigma^2)$. On veut minimiser $L(\beta_0, \beta_1) = \sum \epsilon_i^2 = \sum (y_i - \beta_0 - \beta_1 x_i)^2$. Solution : $\hat{\beta_1} = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}$ et $\hat{\beta_0} = \bar{y} - \hat{\beta_1}\bar{x}$.
Modèle linéaire multiple
$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. On veut minimiser $L(\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||^2 = \frac{1}{2}\sum_i (y_i - \beta_0 - \sum_j \beta_j x_{i,j})^2$. Si $\mathbf{X}^T\mathbf{X}$ est inversible, alors la solution est $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{X}^\dagger\mathbf{Y}$ où $\mathbf{X}^\dagger$ est la pseudo-inverse de Moore-Penrose de $\mathbf{X}$. On a alors $\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}\mathbf{X}^\dagger\mathbf{Y} = \mathbf{P}\mathbf{Y}$ avec $\mathbf{P}$ le projecteur sur l'espace généré par les variables de $\mathbf{X}$.
Modèle non linéaire
$\mathbf{Y} = \boldsymbol{\Phi}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ avec $\boldsymbol{\Phi}$ la matrice où les observations $\mathbf{x_i}$ sont remplacées par $\phi(\mathbf{x_i})$.
Approximation polynomiale : $\phi(\mathbf{x}) = [1, \mathbf{x}, \mathbf{x}^2, \ldots, \mathbf{x}^{d^*}]$ avec d^* le degré du polynôme voulu.
Approximation gaussienne : $\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|^2}{2s^2}\right)$, où $\boldsymbol{\mu}_j$ sont les centres et s est l'écart-type.
Régression régularisée
L'objectif est de lisser la régession pour éviter l'overfitting. La fonction objectif est transformée de la manière suivante : $\mathcal{L}_q(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \alpha\|\boldsymbol{\beta}\|_q^q$ où α est le paramètre de régularisation et $\|\boldsymbol{\beta}\|_q = \left(\sum_{i=1}^d |\beta_i|^q\right)^{1/q}$.
Lasso regression : on prend $q = 1$. Ce lissage change les poids des coefficients de manière à ce que certains deviennent négligeables. Il permet donc de savoir quelles variables ne sont pas très utiles, et ainsi pouvoir réduire la dimention pour optimiser la complexitée.
Ridge regression : on prend $q = 2$. Ce lissage ne vas pas rendre en général des variables négligeables devant d'autres. Cependant, le résultat sera un peu plus proche du résultat sans lissage, ce qui permet de mieux tendre vers la solution.