

IFT3395/6390

Fondements de l'apprentissage machine

Apprentissage non-supervisé

**Réduction de dimensionnalité.
Modèles à variables latentes continues.**

Pascal Vincent 2008

Mise en contexte

Apprentissage supervisé = classification, régression

Apprentissage non supervisé = Algorithmes pour lesquels on ne distingue pas de “cible” explicite dans les données d’entraînement.

- Estimation de densité
(ex: mélange de Gaussiennes)

$$\hat{p}(\mathbf{x})$$

- Clustering, partitionnement
(ex: k-moyennes)

“classification non-supervisée”

- Réduction de dimensionalité...

La réduction de dimensionalité

Qu'est-ce que c'est?

$$\mathbf{z} \in \mathbb{R}^M$$

$$M < D$$

(0.32, -1.3, 1.2)



$$\mathbf{x} \in \mathbb{R}^D$$

(3.5, -1.7, 2.8, -3, 5, -1.4, 2.4, 2.7, 7.5, -3, -2)

La réduction de dimensionnalité

A quoi ça peut servir?

- **Compression** de données (avec perte)
- **Visualisation** des données en 2D ou 3D
- **Extraction de caractéristiques**
potentiellement +fondamentales, +explicatives, +compactes
Prétraitement => meilleure représentation de départ pour un autre algorithme (classification ou régression).

Les algorithmes

Modèles linéaires Gaussiens

- L'Analyse en Composantes Principales (ACP ou PCA) traditionnelle
- L'ACP probabiliste
- L'analyse de facteurs (*factor analysis*)

Modèles non linéaires ou non Gaussiens

- L'ACP à Noyau (*Kernel PCA*)
- L'Analyse en Composantes Indépendantes (ICA)
- Réseaux de neurones auto-associeurs
- Modélisation de variétés (*manifold*) non-linéaires

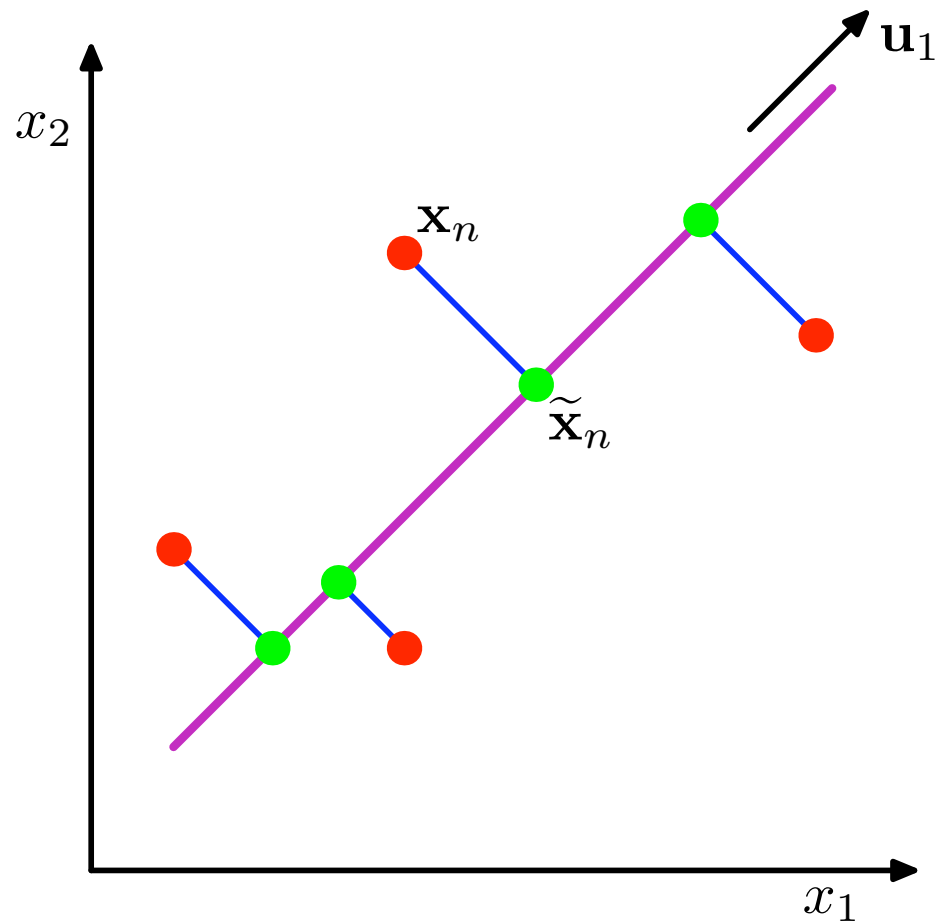
L'ACP classique

- L'ACP trouve un **sous-espace linéaire** qui passe proche des données: **projection orthogonale** de $\mathbf{x} \in \mathbb{R}^D$ sur un **sous-espace linéaire** de plus faible dimension M .
- Les composantes z représentent les coordonnées de la projection de x dans ce sous-espace de dimension M .
- **Très utilisé** comme pré-traitement (extraction de caractéristiques) ou pour la visualisation.
- Un vieil algorithme classique, 2 formulations équivalentes:
 - **minimisation de l'erreur** de reconstruction (*Pearson 1901*)
 - **maximisation de la variance** (*Hotelling 1933*).
- La réinterprétation en tant que modèle probabiliste à variables latentes est beaucoup plus récente.

ACP: deux formulations équivalentes

On cherche les **directions principales \mathbf{u}** : une base orthonormale du sous-espace sur lesquelles projeter les **\mathbf{x}**

- **Minimum d'erreur:**
on minimise la moyenne des distances carrées entre les **\mathbf{x}** et leur **projection** (lignes bleues).
- **Maximum de variance:**
on maximise la variance le long de la direction de projection (variance des **points verts**)



ACP par maximisation de variance

Soit la moyenne empirique $\bar{\mathbf{x}} = \sum_{n=1}^N \mathbf{x}_n$

La variance des points projetés dans la direction \mathbf{u}_1 est:

$$\frac{1}{N} \sum_{n=1}^N (\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}})^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$$

où \mathbf{S} est la matrice de covariance empirique.

Maximiser sous la contrainte $\|\mathbf{u}_1\|^2 = \mathbf{u}_1^T \mathbf{u}_1 = 1$
(multiplicateur de Lagrange) donne:

$$\begin{aligned} \mathbf{S} \mathbf{u}_1 &= \lambda_1 \mathbf{u}_1 \\ \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 &= \lambda_1 \end{aligned}$$

donc la variance est maximale quand \mathbf{u}_1 est le vecteur propre correspondant à la plus grande valeur propre λ_1 .

ACP par maximisation de variance

- Une fois trouvée la première **direction principale**, on peut trouver incrémentalement la 2ème et ainsi de suite, en se restreignant aux directions orthogonales à toutes les précédentes.
- La formulation par minimisation d'erreur donne les mêmes directions principales.

ACP: procédure générale simple

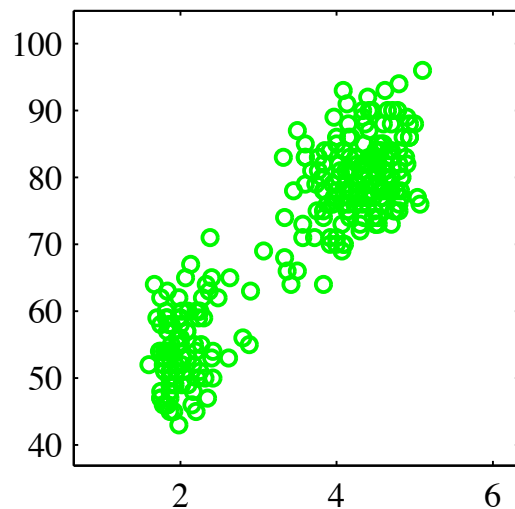
- La base orthonormale \mathbf{U} qu'on obtient est constituée des M premiers vecteurs propres de la matrice de covariance empirique (ceux correspondant aux M plus grandes valeurs propres).
- Le prétraitement par ACP, permet d'obtenir des composantes "normalisées" c.a.d. décorréllées et de variance unitaire:

$$\mathbf{z}_n = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}}) \quad \mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_M \end{pmatrix}$$

- On peut ainsi projeter sur les **directions principales** pour fins de visualisation 2D ou 3D.

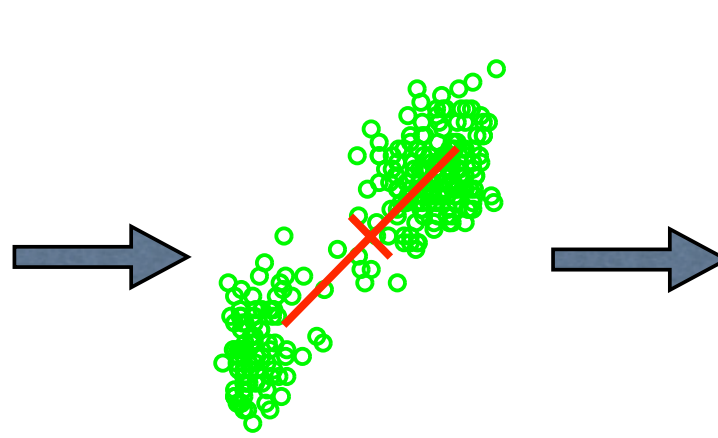
ACP et normalization

données
de départ

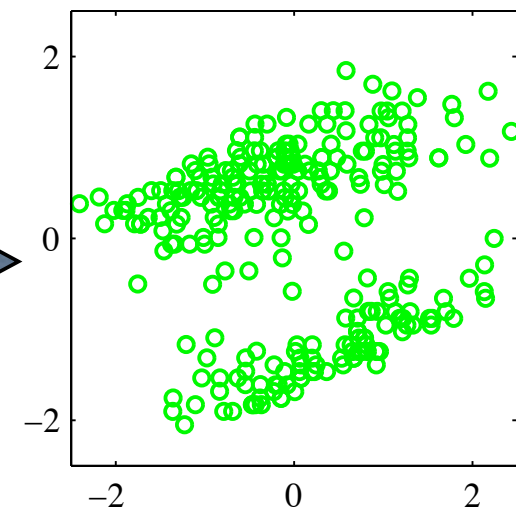


directions
principales

(intervalle $\pm\lambda_i^{1/2}$)



données
prétraitées
(cov = I)

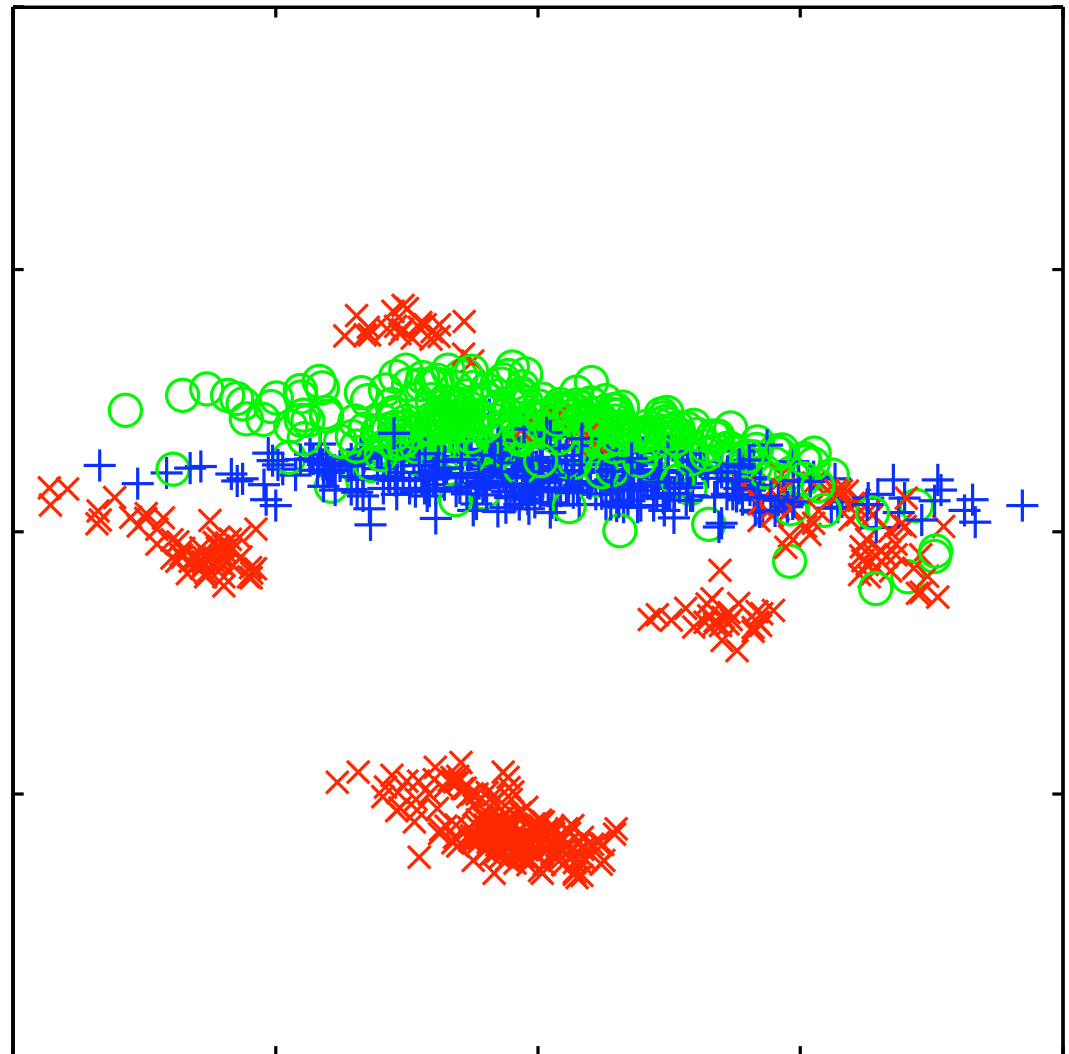


On aurait pu aussi ne conserver que la première composante principale.

ACP pour visualisation

2 composantes principales

Données de départ
en haute dimension



ACP à Noyau (Kernel PCA)

- La transformation apprise par l'ACP est linéaire. Une **version non-linéaire** peut s'obtenir en appliquant l'ACP sur les données transformées par une transformation non linéaire ϕ .

- Les données ainsi transformées ont une matrice de covariance

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T$$

- Et les vecteurs propres dans l'espace transformé sont donnés par $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ et peuvent s'exprimer sous la forme

$$\mathbf{v}_i = \sum_{n=1}^N a_{in} \phi(\mathbf{x}_n)$$

- Avec un noyau k , on peut utiliser **l'astuce du noyau** pour calculer les produits scalaires dans l'espace transformé **sans jamais avoir à faire la transformation explicitement**:

$$\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

ACP à Noyau (Kernel PCA)

- L'astuce du noyau permet de trouver les vecteurs de coefficients $\mathbf{a}_i = (a_{1i}, \dots, a_{ni})^T$ représentant les vecteurs propres en solutionnant le problème de vecteurs propres suivant

$\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i$ où \mathbf{K} est la matrice de Gram:

$$\mathbf{K}_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$$

- On peut ensuite facilement calculer les projections donnant les composantes principales

$$z_i(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{v}_i = \sum_{n=1}^N a_{in} \phi(\mathbf{x})^T \phi(\mathbf{x}_n) = \sum_{n=1}^N a_{in} k(\mathbf{x}, \mathbf{x}_n)$$

- Remarque: si on veut d'abord centrer les données transformées, il faut utiliser une matrice de Gram corrigée (voir Bishop 12.3)

Inconvénients de l'ACP à Noyau

- Il faut calculer la décomposition en vecteurs et valeurs propres d'une matrice de gram $N \times N$ plutôt que d'une matrice $D \times D$. Or généralement $N \gg D$.
=> Peu utilisé en pratique pour de vraies données car trop coûteux.
- Si on conserve seulement les quelques premières composantes principales, on ne peut pas trouver de point correspondant (la projection sur l'hyperplan) dans l'espace de départ.

Explication: le point projeté sur l'hyperplan existe bien dans l'espace transformé, mais n'a pas de pré-image correspondante dans l'espace de départ.

Modèles à variables latentes continues ???

(Bishop, chap. 12)

- **variable observée** (visible): $\mathbf{x} \in \mathbb{R}^D$
- **variable latente** (cachée): $\mathbf{z} \in \mathbb{R}^M$
- à chaque \mathbf{x} correspond un \mathbf{z} différent
(contrairement aux paramètres du modèle, les mêmes pour tout \mathbf{x})
- on peut voir \mathbf{z} comme étant une “*explication*” de \mathbf{x}

Vous avez déjà vu un modèle à variable latente **discrète**
(\mathbf{z} discret) ... Lequel?

- Le **mélange de Gaussiennes!**
- \mathbf{z} indique... à quel groupe (cluster) \mathbf{x} appartient.

ACP probabiliste: un modèle génératif simple

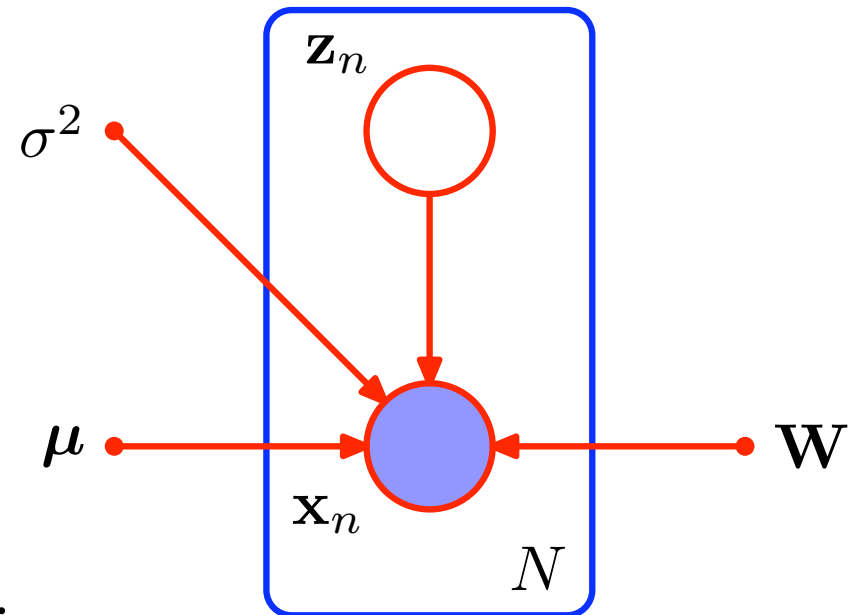
- $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_M)$
- $\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon$

avec $\mu \in \mathbb{R}^D$,

\mathbf{W} une matrice $D \times M$,

$\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_D)$

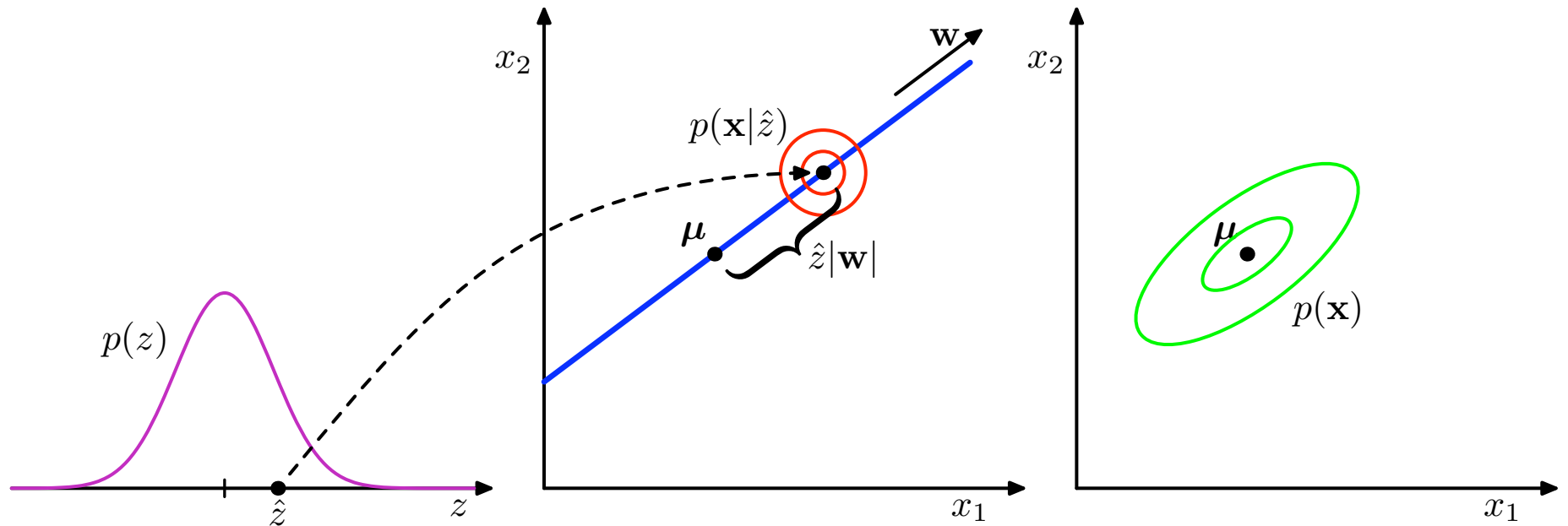
et \mathbf{I}_D la matrice identité $D \times D$.



Cela correspond à $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \sigma^2 \mathbf{I}_D)$

Contrairement à la PCA classique, on définit ainsi un vrai modèle de densité $p(\mathbf{x})$.

ACP probabiliste: illustration du processus générateur



- \mathbf{W} spécifie un ensemble de directions, μ est la moyenne des données et le vecteur de variables latentes \mathbf{z} indique de combien on doit se déplacer à partir de la moyenne dans chaque direction.
- \mathbf{x} est finalement généré en y ajoutant un bruit Gaussien sphérique.

Apprentissage des paramètres du modèle d'ACP probabiliste

Soit par maximisation directe de la vraisemblance

- On peut apprendre les paramètres en maximisant la vraisemblance du modèle étant donné l'ensemble de données \mathbf{X} :

$$\log p(\mathbf{X}|\mathbf{W}, \mu, \sigma^2) = -\frac{N}{2} [D \log(2\pi) + \log |\mathbf{C}| + \text{Tr}(\mathbf{C}^{-1}\mathbf{S})]$$

avec $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$.

- Ceci admet une solution analytique: $\mathbf{W} = \mathbf{U}(\mathbf{\Lambda} - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$
- \mathbf{R} est une matrice de rotation arbitraire. Donc \mathbf{W} n'est la projection de la PCA classique (avec prise en compte des variances) qu'à une rotation près.
- Donc l'ACP probabiliste trouve le **même sous-espace** principal que la PCA classique, mais **pas nécessairement les mêmes composantes**.

Soit avec l'algorithme EM (voir Bishop)

Analyse de facteurs (factor analysis, FA)

- L'Analyse de Facteurs correspond à un modèle génératif très semblable à la PCA probabiliste.
- La seule différence est que le bruit Gaussien final ajouté est diagonal plutôt que sphérique.

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \Psi) \quad \Psi = \begin{pmatrix} \psi_1 & & 0 \\ & \ddots & \\ 0 & & \psi_D \end{pmatrix}$$

- C'est donc un modèle un peu plus riche que la PCA probabiliste (qui correspond au cas particulier $\Psi = \sigma^2 \mathbf{I}_D$)

Comment choisir le nombre de composantes?

- Soit en se basant sur la log vraisemblance obtenue sur un ensemble de validation.
- Soit en adoptant une approche Bayésienne en précisant des distributions à priori pour chacun des paramètres $\mu, \mathbf{W}, \sigma^2$ (voir Bishop 12.2.3)

L'Analyse en Composantes Indépendantes (Independent Component Analysis ICA)

ICA vu comme modèle génératif à variables latentes continues

- Similaire à ACP probabiliste ou analyse de facteurs, mais la distribution sur les variables latentes (les composantes) est non Gaussienne, et factorielle, ce qui correspond à avoir des composantes indépendantes:

$$p(z) = \prod_{j=1}^M p(z_j)$$

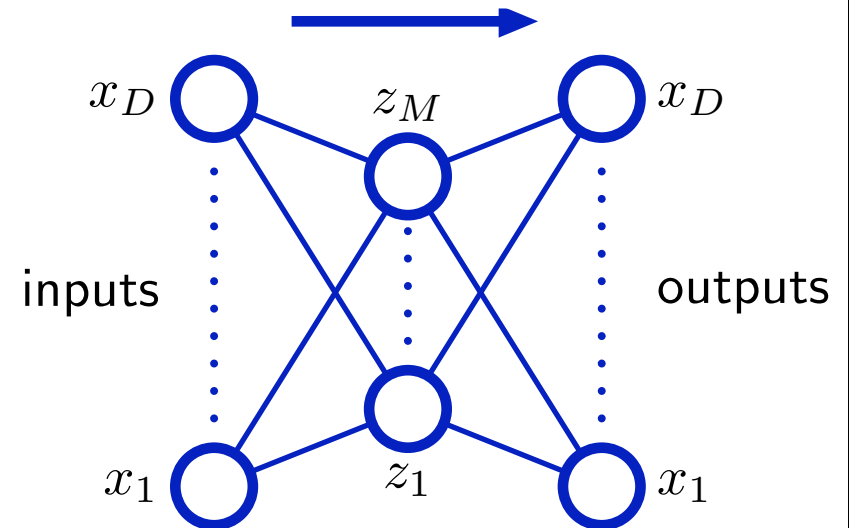
- Tout comme dans ACP probabiliste, les observations \mathbf{x} résultent d'une transformation linéaire des \mathbf{z} (+ bruit)

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon \quad (\text{dans la version originale, } M=D \text{ et } \epsilon = 0, \quad \mu = 0)$$

Il y a beaucoup d'autres façons de voir ICA, notamment comme la maximization de l'information mutuelle $I(\mathbf{x}, \mathbf{z})$.

Réseaux de neurones auto-associateurs

- Un réseau de neurones de type MLP est entraîné à reproduire son entrée (cible=observations)
- La couche cachée est choisie de dimension $M < D$. Ceci entraîne des **erreurs de reconstruction**, que l'entraînement cherche à minimiser.

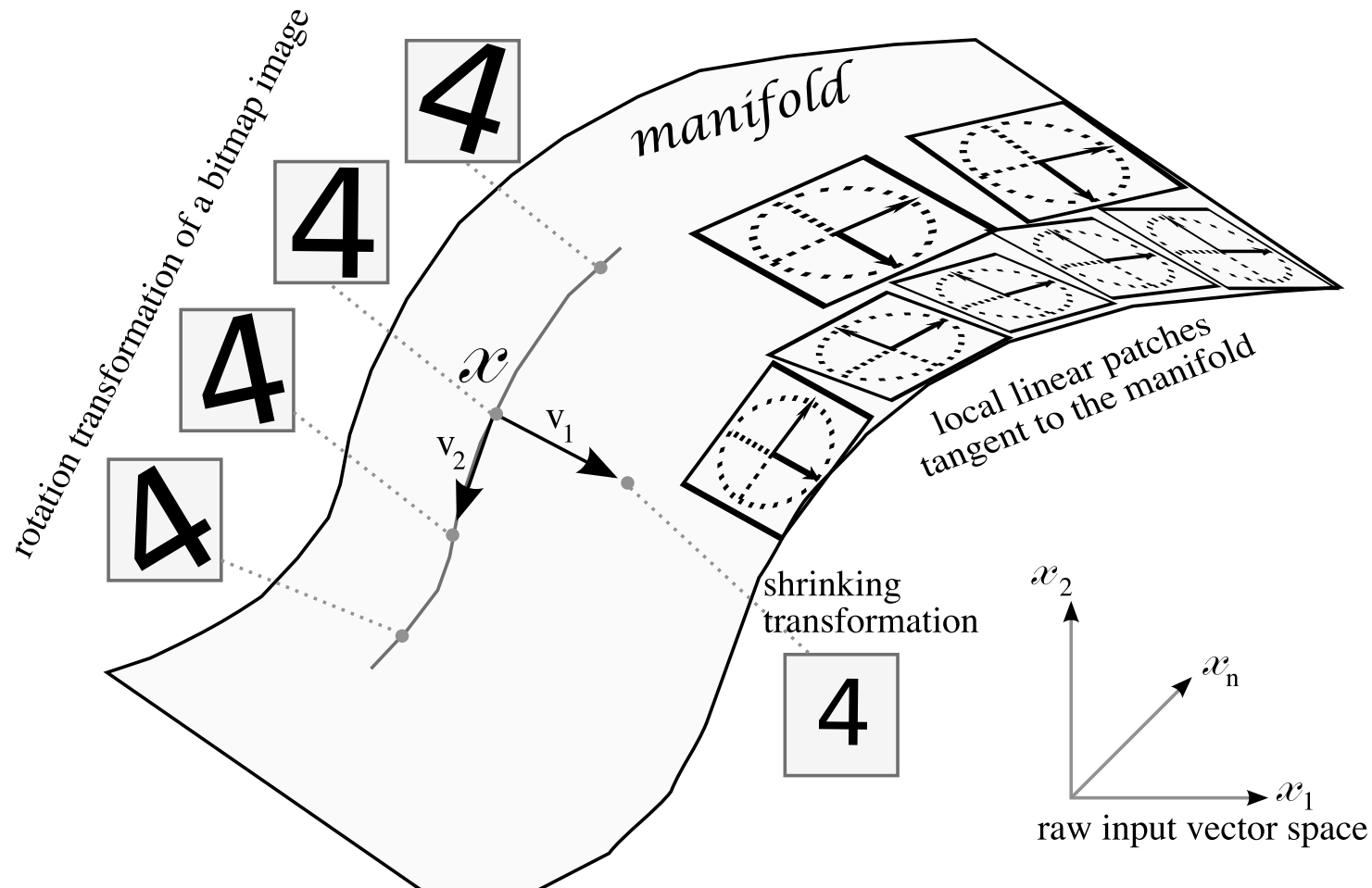


- On obtient une **représentation de dimension réduite** au niveau de la couche cachée.
- Si le réseau est linéaire, ou s'il n'a qu'une couche cachée (même non-linéaire) c'est équivalent à la PCA.
- S'il y a plusieurs couches cachées avec des non-linéarités c'est une **méthode de réduction de dimensionalité non-linéaire**.

L'hypothèse de variété (*manifold hypothesis*)

une motivation géométrique pour la réduction de dimensionalité

degrés de variabilité dans les données semble $< D$



Modélisation de variétés

- On peut modéliser une variété non-linéaire par un mélange de modèles linéaires, par ex. un mélange d'analyses de facteurs.
- Autres méthodes de réduction de dimensionnalité:
 - ▶ Multidimensional Scaling (MDS)
 - ▶ Locally Linear Embedding (LLE)
 - ▶ Isometric Feature Mapping (Isomap)
 - ▶ ...

Questions ouvertes

- Pour apprendre une bonne représentation: **réduction de dimensionalité ou augmentation (+sparsité) ?**
- ...