Apprentissage Statistique
Partie 2

Sorbonne Université- Master DAC et Master Math Spécialité Big Data - P. Gallinari,
patrick.gallinari@lip6.fr, http://www-connex.lip6.fr/~gallinar/

nom.prenom@lip6.fr

Année 2018-2019

Apprentissage supervisé

Machines à noyaux

Introduction

- Familles de machines d'apprentissage générales qui exploitent l'idée suivante :
  - Projeter les données dans un espace de grande dimension - éventuellement infini -où le problème sera facile à traiter
  - Utiliser des "projections" non linéaires permettant des calculs "efficaces"
- Exemples :
  - Machines à Vecteurs Support (généralisent : hyperplan optimal, cadre Vapnik)
  - Processus Gaussien (généralisent : régression logistique, cadre Bayesien)
- Cours
  - Machines à Vecteurs Support

Perceptron : formulation duale

hyp: 2 classes linéairement séparables, sorties désirées -1, 1

Perceptron primal

Initialiser $\boldsymbol{w}(0) = 0$

Répeter (t)

choisir un exemple, $(\boldsymbol{x}(t), d(t))$

si $d(t)\boldsymbol{w}(t).\boldsymbol{x}(t) \leq 0$

alors $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + d(t)\boldsymbol{x}(t)$

Jusqu'à convergence

Fonction de décision

$$F(x) = sgn(\sum_{j=0}^{n} w_j x_j),$$

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \boldsymbol{d^i x^i}$$

α_i : nombre de fois où l'algorithme a rencontré une erreur de classification sur $\boldsymbol{x}^i$

Perceptron dual

Initialiser $\boldsymbol{\alpha} = \boldsymbol{0}, \boldsymbol{\alpha} \in \boldsymbol{R^N}$

Répeter (t)

choisir un exemple, $(\boldsymbol{x}(t), d(t))$

soit k: $\boldsymbol{x}(t) = \boldsymbol{x}^k$

si $d(t)\sum_{i=1}^{N} \alpha_i \boldsymbol{d^i x^i}.\boldsymbol{x}(t) \leq 0$

alors $\alpha_k = \alpha_k + 1$

Jusqu'à convergence

Fonction de décision

$$F(x) = sgn(\sum_{i=1}^{N} \alpha_i \boldsymbol{d^i x^i}.\boldsymbol{x})$$

Matrice de Gram G :

matrice NxN de terme i, j : $\boldsymbol{x}^i.\boldsymbol{x}^j$

matrice de similarité entre données

Représentation Duale

- La plupart des machines à apprentissage linéaires ont une représentation duale
  - Exemples
    - Adaline, regression, regression ridge, etc
  - L'information sur les données est entièrement fournie par la matrice de Gram : $G = \left(x^i . x^j\right)_{i,j=1\ldots N}$, qui joue un rôle central
  - La fonction de décision $F(x)$ s'exprime comme une combinaison linéaire de produits scalaires entre la donnée d'entrée *x* et les exemples d'apprentissage

- Les machines à noyau généralisent ces idées
  - Une fonction noyau K est définie sur $R^n \mathrm{x} R^n$ (on suppose $x \in R^n$) par

$$K(x, z) = < \Phi(x), \Phi(z) >$$

où Φ est une fonction de R^n dans un espace muni d'un produit scalaire

Produit Scalaire et Noyaux

- Projection non linéaire dans un espace de grande dimension H
  - (en général dim H >> n, et peut même être infinie)
  - $\Phi: R^n \rightarrow R^p$ avec $p \gg n$
  - Machine linéaire dans H - Primal :
    - $F(x) = \sum_{j=1}^{p} w_j \Phi_j(x) + b$
  - Machine linéaire dans H - Dual :
    - $w = \sum_{i=1}^{N} d^i \alpha_i \Phi(\boldsymbol{x}^i), \qquad \mathrm{F(x)} = \sum_{i=1}^{N} d^i \alpha_i \Phi(\boldsymbol{x}^i).\Phi(\boldsymbol{x}) + b,$
- Calculer les produits scalaires dans l'espace initial : choisir F /
  - $\Phi(x).\Phi(x') = K(x, x')$
  - $\mathrm{F(x)} = \sum_{i=1}^{N} d^i \alpha_i K(x^i, x) + b$
  - avec K : fonction noyau (i.e. symétrique)

- Généralise le produit scalaire dans l'espace initial
- Le calcul de F ne dépend pas directement de la taille de H : les calculs sont faits dans l'espace initial.
- La machine linéaire dans H peut être construite à partir d'une fonction K sans qu'il soit nécessaire de définir explicitement Φ : en pratique, on spécifiera directement K.
- Cette idée permet d'étendre de nombreuses techniques linéaires au non linéaire: il suffit de trouver des noyaux appropriés
  - Exemples
    - ACP, analyse discriminante, regression, etc

Caractérisation des noyaux

- Quand peut on utiliser cette idée ?
- Cas d'un espace fini
  - Soit $X = \{x^1, \ldots, x^N\}$, $K(x, x')$ une fonction symétrique sur X, K est une fonction noyau ssi la matrice N x N dont l'élément (i, j) est $K(x^i, x^j)$ est positive semi-définie (valeurs propres ≥ 0 ou $x^T \boldsymbol{K} x > 0 \; \forall x$, avec $\boldsymbol{K} = Matrice\left(K\left(x^i, x^j\right); i, j = 1..N\right)$)
- Cas général : Conditions de Mercer (noyaux de Mercer)
  - Il existe une application Φ et un développement

$$K(x, x') = \sum_{i=1}^{\infty} \Phi(x)_i . \Phi(x')_i$$

  - Ssi :

$$\forall g / \int g(x)^2 dx \quad \text{est fini et} \int K(x, x') g(x) g(x') dx dx' \geq 0$$

Caractérisation des noyaux
Espace de Hilbert à noyau autoreproduisant

- Une fonction $K: X * X \rightarrow R$ qui est soit continue soit définie sur un domaine fini peut s'écrire sous la forme d'un produit scalaire :

$$K(x,z) = <\phi(x), \phi(z)>$$

avec $\Phi : x \rightarrow \Phi(x) \in \boldsymbol{F}$ espace de Hilbert

ssi c'est une fonction symétrique et toutes les matrices formées par la restriction de K à un échantillon fini sur *X* sont semi-définies positives).

- Résultat à la base de la caractérisation effective des fonctions noyaux
- Il permet de caractériser *K* comme un noyau sans passer par Φ
- C'est une formulation équivalente aux conditions de Mercer

Caractérisation des noyaux
Espace de Hilbert à noyau autoreproduisant

- L'espace de Hilbert associé au noyau *K* :

$$F = \left\{ \sum_{i=1}^{l} \alpha_i K(x_i,.) \,/\, l \in N, x_i \in X, \alpha_i \in R, i = 1..l \right\}$$

- Le produit scalaire défini sur cet espace :

$$\text{Soient} \quad f(.) = \sum_{i=1}^{l} \alpha_i K(x_i,.), \qquad g(.) = \sum_{j=1}^{n} \beta_j K(x_j,.)$$

$$< f, g > = \sum_{i=1}^{l} \sum_{j=1}^{n} \alpha_i \beta_j K(x_i, z_j) = \sum_{i=1}^{l} \alpha_i g(x_i) = \sum_{j=1}^{n} \beta_j f(z_j)$$

- Noyau auto-reproduisant
  - Si on prend $g(.) = K(x,.)$ alors $< f, K(x,.) > = \sum_{i=1}^{l} \alpha_i K(x_i, x) = f(x)$

Exemples de noyaux

$$K(x,z) = < x.z >^2$$

$$K(x,z) = \left(\sum_{i=1}^{n} x_i.z_i \right)^2 = \sum_{i,j=1}^{n} (x_i.x_j)(z_i.z_j)$$

$$K(x,z) = < \phi(x).\phi(z) > \text{ avec } \phi(x) / \phi(x)_{i,j} = (x_i.x_j)_{i,j=1,n}$$

i.e. tous les monomes de d° 2 : $\begin{pmatrix} n+1 \\ 2 \end{pmatrix}$

$$K(x,z) = (< x.z > + c)^2$$

$$K(x,z) = < \phi(x).\phi(z) > \text{ avec } \phi(x) / \phi(x)_{i,j} = ((x_i.x_j)_{i,j=1,n}, (\sqrt{2}cx_i)_{i=1,n}, c)$$

i.e. ss ensemble des polynomes de d° 2

Exemples de noyaux (suite)

$$K(x, x^i) = \begin{cases} (x.x^i + 1)^d & \Phi \text{ polynome d'ordre d} \\ \exp -\gamma \left\| x - x^i \right\|^2 & \Phi \text{ noyau gaussien} \\ Sigmoïde(vx.x^i + c) & \end{cases}$$

- En pratique, on utilise souvent des noyaux
  - Linéaires
  - Gaussiens
  - Polynomiaux

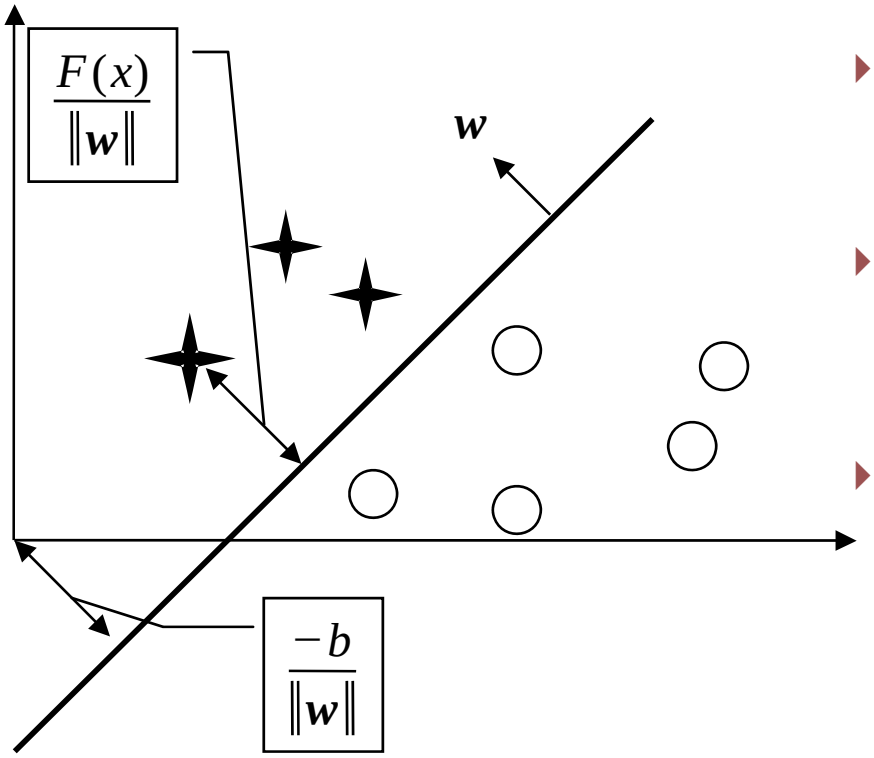
Construction des noyaux en pratique

- Les résultats de Mercer servent à prouver les propriétés des fonctions noyaux. En pratique, elles sont peu utiles
- Pour construire des noyaux, on procède par combinaison à partir de noyaux connus
- Si K_1 et K_2 sont des noyaux sur X^2, K_3 défini sur F, les fonctions suivantes sont des noyaux :
  - $K(x,z) = K_1(x,z) + K_2(x,z)$
  - $K(x,z) = K_1(x,z).K_2(x,z)$
  - $K(x,z) = aK_1(x,z)$
  - $K(x,z) = K_3(\phi(x),\phi(z))$
  - …..

Machines à vecteurs support

- Exposé du cours : discrimination 2 classes
- Cas général : discrimination multi-classes, régression, densité
- Idées
  - Projeter -non linéairement- les données dans un espace de "très" grande taille H
  - Faire une séparation linéaire de bonne qualité dans cet espace
  - Raisonner dans H, mais résoudre le problème d'optimisation dans l'espace de départ (noyaux)

- Notion de marge



- Hyperplan H
  - $F(\boldsymbol{x}) = \boldsymbol{w}.\boldsymbol{x} + b = 0$
- Marge géométrique pour x^i
  - $M(\boldsymbol{x}^i) = d^i \left(\frac{\boldsymbol{w}.\boldsymbol{x}^i}{\|\boldsymbol{w}\|} + \frac{b}{\|\boldsymbol{w}\|}\right) = d^i \frac{F(\boldsymbol{x}^i)}{\|\boldsymbol{w}\|}$
- Marge de $\boldsymbol{w}$ par rapport à un ensemble de données *D*
  - $Min_i M(x^i)$
- Hyperplan de marge maximale
  - $Max_w(Min_i M(x^i))$

Marge géométrique vs marge fonctionnelle

- Marge géométrique
  - $d^i \frac{F(\boldsymbol{x}^i)}{\|\boldsymbol{w}\|}$
- Marge fonctionnelle
  - $d^i F(x^i)$
- Remplacer $\boldsymbol{w}$ par $k\boldsymbol{w}$ ne change pas la fonction de décision ou la marge géométrique, mais change la marge fonctionnelle.
- Pour les SVM, on fixera la marge fonctionnelle à 1 et on optimisera la marge géométrique.

Prémisses : Séparation linéaire à hyperplan optimal (1974)

- Hyp : *D* linéairement séparable

- Fonction de décision : $F(x) = w.x + b$ $\quad D=\{x^i,d^i\}_{i=1..N}$ avec $d^i=\pm 1$
- Pb apprentissage :
  - trouver l'hyperplan optimal *H** qui sépare *D* i.e.
    - $d^i.F(x^i) \geq 1, \forall i$

- avec une marge géométrique maximale *M* = $\min_i \frac{d^i.F(x^i)}{\|w\|} = \frac{1}{\|w\|}$

i.e. : Problème Primal :

$$\begin{cases} \text{Minimiser } \|w\|^2 \\ S.C. \quad d^i.F(x^i) \geq 1 \end{cases}$$

- Dans le cas de noyaux linéaires, on résout en général le problème primal
  - Il existe de nombreux algorithmes rapides
- Dans le cas de noyaux non linéaires, on va en général résoudre une version duale du problème
  - On introduit ensuite quelques notions d'optimisation sous contrainte pour décrire la formulation duale du problème

Intermède

Optimisation
Problèmes sous contraintes égalités, inégalités

Optimisation
Problèmes sous contraintes égalités, inégalités

- Soient
  - $f, g_{i,i=1,\dots,k}, h_{j,j=1,\dots,m}$ des fonctions définies sur $\mathbb{R}^n$ à valeur dans $\mathbb{R}$
- On considère le problème primal suivant (pb. (0)) :

$$Min\ f(\boldsymbol{w}), \boldsymbol{w} \in \Omega \subset \mathbb{R}^n$$

Sous contraintes

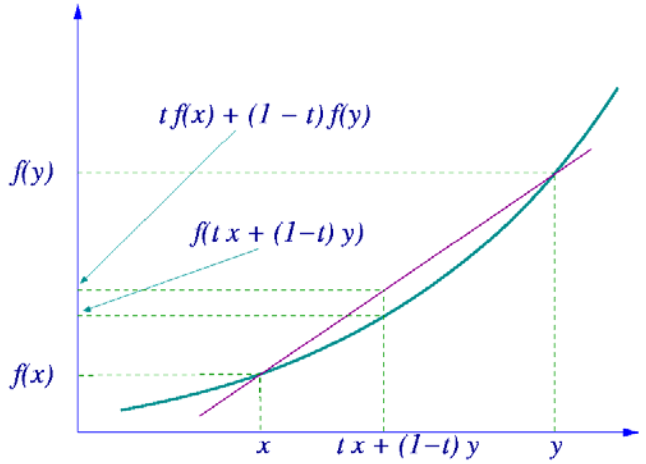
$g_i(\boldsymbol{w}) \leq 0, i = 1, \dots, k$ noté $\boldsymbol{g}(\boldsymbol{w}) \leq 0$

$h_j(\boldsymbol{w}) = 0, j = 1, \dots, m$ noté $\boldsymbol{h}(\boldsymbol{w}) = 0$

  - **Fonction objectif** $f(\boldsymbol{w})$
  - **Région admissible** $R = \{\boldsymbol{w} \in \Omega: \boldsymbol{g}(\boldsymbol{w}) \leq 0, \boldsymbol{h}(\boldsymbol{w}) = 0\}$, région de Ω où f est définie et les contraintes vérifiées
  - $\boldsymbol{w}^*$ est un **minimum global** si il n'existe pas d'autre point $\boldsymbol{w}$ tel que $f(\boldsymbol{w}) < f(\boldsymbol{w}^*)$, c'est un optimum local si $\exists \epsilon > 0: f(\boldsymbol{w}) \geq f(\boldsymbol{w}^*)$, sur la boule $\|\boldsymbol{w} - \boldsymbol{w}^*\| < \epsilon$
  - Une contrainte $g_i(w) \leq 0$ est dite **active** si la solution $\boldsymbol{w}^*$ vérifie $\boldsymbol{g}(\boldsymbol{w}^*) = 0$ et inactive sinon
  - La valeur optimale de la fonction objectif (solution du pb. (0) est appelée la valeur du problème d'optimisation primal)

Optimisation - Fonctions convexes

- $f(w)$ est convexe pour $\boldsymbol{w} \in \mathbb{R}^n$si
- $\forall t \in [0,1], \forall \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n, \forall \mathrm{t} \in [0,1], \quad f(t\boldsymbol{w} + (1-t)\boldsymbol{v}) \leq tf(\boldsymbol{w}) + (1-t)f(\boldsymbol{v})$



- Un ensemble $\Omega \subset \mathbb{R}^n$est convexe si $\forall \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n, \forall \mathrm{t} \in [0,1], t\boldsymbol{w} + (1-t)\boldsymbol{v} \in \Omega$
- Si une fonction est convexe, tout minimum local est un minimum global
- Un problème d'optimisation pour lequel Ω est convexe, la fonction objectif et les contraintes sont convexes est dit convexe

Optimisation non contrainte

- Th. Fermat
  - Une Condition Nécessaire pour que w* soit un min. de $f(\boldsymbol{w})$, $f \in C^1$ est $\frac{\partial f(\boldsymbol{w}^*)}{\partial w} = 0$
  - Si f est convexe c'est une Condition Suffisante

Optimisation avec contraintes égalités
Lagrangien

- Optimisation avec contraintes égalité (pb (1)):

$$Min\ f(\boldsymbol{w}), \boldsymbol{w} \in \Omega \subset \mathbb{R}^n$$

Sous les contraintes

$$h_j(\boldsymbol{w}) = 0, \quad j = 1, \ldots, m \qquad \text{noté } \boldsymbol{h}(\boldsymbol{w}) = 0$$

- On définit le Lagrangien $L(\boldsymbol{w}, \boldsymbol{\beta})$ associé à ce problème par

$$L(\boldsymbol{w}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{j=1}^{m} \beta_j h_j(\boldsymbol{w})$$

  - les β_j sont les coefficients de Lagrange
- Rq.
  - Si $\boldsymbol{w}^*$ est une solution du problème d'optimisation sous contrainte, Il est possible que $\frac{\partial f(\boldsymbol{w}^*)}{\partial w} \neq 0$

Optimisation avec contraintes égalités
Th. Lagrange

- Th. Lagrange
  - Une condition necessaire pour que $\boldsymbol{w}^*$, soit solution de (pb. (I)), avec $f, h_i \in C^1$ est
    - $\frac{\partial L(\boldsymbol{w}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{w}} = 0$
    - $\frac{\partial L(\boldsymbol{w}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{\beta}} = 0$
  - Si $L(\boldsymbol{w}, \boldsymbol{\beta}^*)$est une fonction convexe de $\boldsymbol{w}$, c'est une condition suffisante
  - Rq
    - La première condition donne un nouveau système d'équations
    - La seconde donne les contraintes

Optimisation sous contraintes égalité + inégalités - Lagrangien augmenté

- De même, on définit le Lagrangien augmenté pour le pb. (0) :

$$L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{j=1}^{m} \beta_j h_j(\boldsymbol{w})$$

- Rappel : pb. (0)

$Min\ f(\boldsymbol{w}), \boldsymbol{w} \in \Omega \subset \mathbb{R}^n$

Sous contraintes

$g_i(\boldsymbol{w}) \leq 0, i = 1, \ldots, k$ noté $\boldsymbol{g}(\boldsymbol{w}) \leq 0$

$h_j(\boldsymbol{w}) = 0, j = 1, \ldots, m$ noté $\boldsymbol{h}(\boldsymbol{w}) = 0$

Optimisation – formulation duale

- Formulation duale du problème d'optimisation
  - Le problème d'optimisation dual correspondant au problème primal pb (0) est :

$$Maximiser_{\alpha,\beta} \ \theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{w \in \Omega} L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\text{sous contrainte } \boldsymbol{\alpha} \geq 0$$

    - Max $\theta(\boldsymbol{\alpha}, \boldsymbol{\beta})$ est appelé la valeur du dual
    - Rq : $\min_{w \in \Omega} L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ est une fonction de $\boldsymbol{\alpha}, \boldsymbol{\beta}$ uniquement
  - Propriété (dualité faible) : la valeur du dual est bornée supérieurement par la valeur du primal

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha} \geq 0} \ \min_{w \in \Omega} L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \leq \min_{w \in \Omega} \ \max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha} \geq 0} \ L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Dans certains cas, on a égalité, cf. dualité forte

Optimisation : dualité faible

- $\max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha}\geq 0} \min_{\boldsymbol{v}\in\Omega} L(\boldsymbol{v},\boldsymbol{\alpha},\boldsymbol{\beta}) \leq \min_{\mathbf{w}\in\Omega} \max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha}\geq 0} L(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta})$:
  - $\theta_D(\alpha,\beta) = \min_{v\in\Omega} L(\boldsymbol{v},\boldsymbol{\alpha},\boldsymbol{\beta})$:
    - $\theta_D(\alpha,\beta) \leq L(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta})$ pour un $\boldsymbol{w}$ qque admissible
    - $L(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) = f(w) + \alpha.g(w) + \beta.h(w)$ avec $\alpha \geq 0, g(\boldsymbol{w}) \leq 0, h(\boldsymbol{w}) = 0$
    - $L(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) \leq f(w)$
    - $\theta_D(\alpha,\beta) \leq f(w)$
    - $\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\alpha}\geq 0} \min_{v\in\Omega} L(v,\boldsymbol{\alpha},\boldsymbol{\beta}) \leq f(w)$ pour un w qque admissible
  - $\theta_P(\boldsymbol{w}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha}\geq 0} L(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta})$
    - $\theta_P(\boldsymbol{w}) = f(\boldsymbol{w}) + \max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha}\geq 0} (\alpha.g(\boldsymbol{w}) + \beta.h(\boldsymbol{w}))$
    - $\theta_P(\boldsymbol{w}) = \begin{cases} f(\boldsymbol{w}) \text{ si } \boldsymbol{w} \text{ est admissible} \\ +\infty \text{ sinon} \end{cases}$ car $\max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha}\geq 0} (\alpha g(w) + \beta h(w)) = 0$ pour un point admissible
    - $\theta_P(\boldsymbol{w}) = f(\boldsymbol{w})$
    - Rq: on retrouve le problème primal original : $\min_{\mathbf{w}\in\Omega} \max_{\boldsymbol{\alpha},\boldsymbol{\beta};\boldsymbol{\alpha}\geq 0} L(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) = \min_{\mathbf{w}\in\Omega} f(\boldsymbol{w})$
- D'où l'inégalité de dualité faible

- Théorème de dualité forte
  - Etant donné un problème d'optimisation

    $Min\ f(\boldsymbol{w}), \boldsymbol{w} \in \Omega \subset \mathbb{R}^n$ convexe et $f \in C^1$ convexe

    Sous contraintes

    $g_i(\boldsymbol{w}) \leq 0, i = 1, \ldots$,k noté $\boldsymbol{g}(\boldsymbol{w}) \leq 0$

    $h_j(\boldsymbol{w}) = 0, j = 1, \ldots$,m noté $\boldsymbol{h}(\boldsymbol{w}) = 0$

    où les g_i et les h_j sont affines ($h_j(\boldsymbol{w}) = A_{\boldsymbol{j}}\boldsymbol{w} + b_j$)

    - alors les valeurs du primal et du dual sont égales
    - Les conditions d'existence d'un optimum sont données par le théorème de Kuhn et Tucker

Optimisation
Th. Kuhn et Tucker

- On considère (pb. (0)) avec Ω convexe et $f \in C_1$ convexe, g_i, h_j affines $(h = A.w + b)$
- Une CNS pour que w* soit un optimum est qu'il existe α* et β* :

$$\begin{cases} \dfrac{\partial L(w^*, \alpha^*, \beta^*)}{\partial w} = 0 \\ \dfrac{\partial L(w^*, \alpha^*, \beta^*)}{\partial \beta} = 0 \\ \alpha_i^* \, g_i(w^*) = 0, i = 1..k \\ g_i(w^*) \le 0, i = 1..k \\ \alpha_i^* \ge 0, i = 1..k \end{cases}$$

- Sous les hypothèses de convexité, la formulation duale est une alternative à la formulation primale qui peut se révéler plus simple à traiter (e.g. SVM non linéaires)

Optimisation

- Rq
  - La 3e condition dite condition complémentaire de Karush-Kuhn-Tucker implique que pour une contrainte active $\alpha_i^* \geq 0$ alors que pour une contrainte inactive $\alpha_i^* = 0$
    - Soit une contrainte est active ($\alpha_i^* \geq 0$ et $g_i(w^*) = 0$), w^* est un point frontière de la région admissible
    - Soit elle est inactive ($\alpha_i^* = 0$) et w^* est dans la région admissible
  - Si le point solution w^* est dans la région admissible (contrainte inactive) alors les conditions d'optimalité sont données par le th. de Fermat et $\alpha_i^* = 0$. Si il est sur la frontière (contrainte active), les conditions d'optimalité sont données par le th. de Lagrange avec $\alpha_i^* > 0$.

- Fin de l'intermède

SVM – formulations primale et duale

Cas d'un noyau linéaire

- SVM
  - Ω, f et les contraintes sont convexes, L est quadratique
  - On étudie le cas, $D = \{(x^i, d^i)\}_{i=1...N}$ linéairement séparables, $d^i \in \{-1,1\}$
- Pb. Primal

$$Min(w.w) \quad \text{(i.e. max la marge)}$$

sous les contraintes

$$d^i(w.x^i + b) \geq 1, i = 1..N$$

- Lagrangien primal

$$L(w,b,\alpha) = \frac{1}{2} w.w - \sum_{i=1}^{N} \alpha_i (d^i (w.x^i + b) - 1)$$

- Lagrangien dual

$$L(w,b,\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} d^i d^j \alpha_i \alpha_j (x^i.x^j)$$

- Avec $\alpha_i \geq 0$ dans les 2 cas

SVM – formulations primale et duale

- Pb. Dual

$$\text{Max} \quad L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} d^i d^j \alpha_i \alpha_j (x^i . x^j)$$

$$\text{sous les contraintes}$$

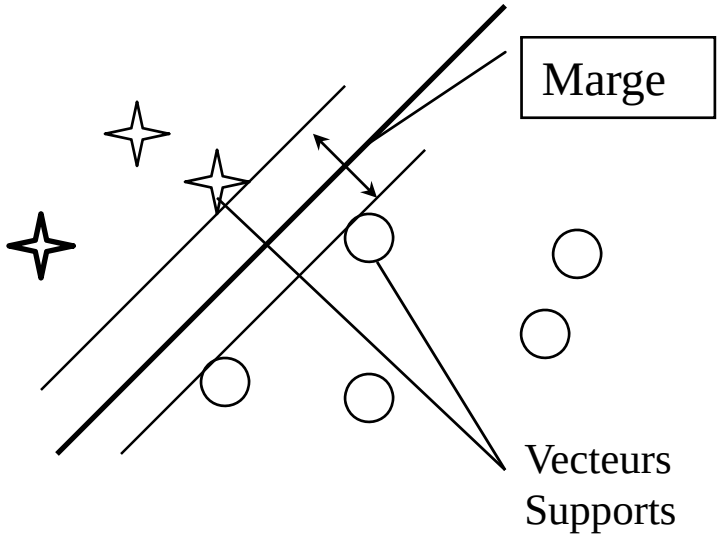
$$\begin{cases} \sum_{i=1}^{N} d^i \alpha_i = 0 \\ \alpha_i \geq 0, i = 1..N \end{cases}$$

- C'est un problème d'optimisation quadratique sous contrainte

- Solution : $\boldsymbol{w}^*$ dépend uniquement des points supports, i.e. des points sur la marge qui vérifient : $d^i F^*\left(x^i\right) = 1$
- la fonction de décision prend la forme

$$F(x, \alpha^*, \beta^*) = \sum_{i \, vecteur \, \text{sup} \, port} d^i \alpha^*_i \, (x^i . x) + b^*$$

  - Rq: Quelque soit la dimension de l'espace, le nombre de degrés de liberté est "égal" au nombre de points de support
  - *F** dépend uniquement du produit scalaire $x^i . x$



Machines à vecteurs supports
cas de noyaux non linéaires

- Faire une séparation à marge max. dans un espace défini par une fonction noyau.
- Tous les résultats sur le classifieur linéaire à marge max. se transposent en remplaçant $x^i.x$ par $K(x^i, x)$

$$\Phi : R^n \rightarrow R^p$$

$$W = \sum_{x^i \, V.S.} d^i \alpha_i \Phi(x^i) \qquad F(x) = \sum_{x^i \, V.S.} d^i \alpha_i \Phi(x^i) \Phi(x) + b$$

$$\Phi(x).\Phi(x') = K(x, x')$$

$$F(x) = \sum_{x^i \, V.S.} d^i \alpha_i K(x, x^i) + b$$

Machines à vecteurs supports

- Apprentissage :
  - On résout le problème d'optimisation dual :

$$\text{Maximiser} \quad L(\alpha) = \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j d^i d^j K(x^i, x^j)$$

$$S.C \quad \alpha_i \geq 0 \quad \text{et} \quad \sum_i \alpha_i d^i = 0$$

  - Problème minimisation quadratique sous contraintes dans l 'espace de départ
  - Difficile en pratique : différents algorithmes.
  - Dans la solution optimale $\alpha_i > 0$ uniquement pour les points support.
    - Seuls les produits scalaires *K* apparaissent, et pas les Φ.

Propriétés de généralisation -exemples

- Th 1 $$E\,[P\,(erreur(x\,))] \le \frac{E\,[\#\,\text{vecteurs supports}]}{\#\,\text{exemples apprentissage} - 1}$$
  - peu de points support → meilleure généralisation
  - indépendant de la taille de l'espace de départ
- Th 2
  - Si $$\exists\, q/\forall i \;= 1..N\;,\; \left\| x^{\,i} \right\| \le q$$
  - l'hyperplan optimal passe par l'origine et a pour marge ρ
  - Alors $$E\,[P\,(erreur(x\,))] \le \frac{E\,[\frac{q}{2}^{2}]}{\rho^{2}}$$

    - Dans les 2 cas, E[P()] est l'espérance sur tous les ensembles de taille l-1, et E[membre droit] est l'espérance sur tous les ensembles d'apprentissage de taille l (leave one out).

Machines à vecteurs supports
Cas non linéairement séparable

- Marges molles
  - L'algorithme est instable
    - Dans les cas non linéairement séparables
    - Dans le cas de données réelles même linéairement séparables
    - **Solution adoptée en pratique**
      - autoriser des erreurs, i.e. prendre pour contraintes :

$$d^i (W \cdot \Phi(x^i) + b) \geq 1 - \eta^i$$

$$\eta^i \geq 0$$

- $\eta^i = 0, x^i$ est correctement classifié et est du bon coté de la marge
- $0 < \eta^i <= 1, x^i$ est correctement classifié, est à l'intérieur de la marge
- $\eta^i > 1, x^i$ est mal classé
- η^i : *slack variable*

Machines à vecteurs supports
Cas non linéairement séparable

- But
  - Maximiser la marge tout en pénalisant les points qui sont mal classés
- Formalisation
  - Plusieurs expressions possibles du problème
  - L'une des plus courantes :

$$Min(w.w) + C\sum_{i=1}^{N} \eta^{\mathrm{i}} \qquad \text{(i.e. max la marge)}$$

$$S.C.$$

$$d^i(w.x^i + b) \geq 1 - \eta^{\mathrm{i}}, i = 1..N$$

$$\eta^{\mathrm{i}} \geq 0, i = 1..N$$

  - C fixé par validation croisée joue le rôle de paramètre de régularisation

Machines à vecteurs supports
Cas non linéairement séparable

- Marges molles – formulation duale

Maximiser $$L(\alpha) = \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j d^i d^j K(x^i, x^j)$$

$$S.C \quad 0 \le \alpha_i \le C \quad \text{et} \quad \sum_i \alpha_i d^i = 0$$

Algorithmes d'optimisation

- **Algorithmes d'optimisation standard pour la programmation quadratique sous contrainte**
  - **e.g. Sequential Minimal Optimization (SMO)**
- **Algorithmes stochastiques - SVM Results –(Bottou 2007)**
  - Task : Document classification - RCV1 documents belonging to the class CCAT (2 classes classification task)
    - Programs SVMLight and SVMPerf are well known SVM solvers written by Thorsten Joachims. SVMLight is suitable for SVMs with arbitrary kernels. Similar results could be achieved using Chih-Jen Lin's LibSVM software. SVMPerf is a specialized solver for linear SVMs. It is considered to be one of the most efficient optimizer for this particular problem.

Algorithm (hinge loss)	Training Time	Primal cost	Test Error
SVMLight	23642 secs	0.2275	6.02%
SVMPerf	66 secs	0.2278	6.03%
Stochastic Gradient (svmsgd)	**1.4 secs**	0.2275	6.02%
Stochastic Gradient (svmsgd2	**1.4 secs**	0.2275	6.01%

Apprentissage non supervisé

Algorithme EM et mélange de densités
Spectral clustering
Non Negative Matrix Factorization

Applications

- analyse des données quand il n'y a pas de connaissance sur la classe.
  - e.g. pas d'étiquetage des données (problème nouveau)
- trop de données ou étiquetage trop compliqué
  - e.g. traces utilisateur (web), documents web, parole, etc
- réduction de la quantité d'information
  - e.g. quantification
- découverte de régularités sur les données ou de similarités.

Apprentissage non supervisé

Algorithme Espérance Maximisation (EM)

Application aux mélanges de densités

Algorithme E. M. (Espérance Maximisation) - Introduction

- On dispose
  - de données $D = \{x^i; i = 1 \dots N\}$
    - On n'a pas d'étiquette d^i associée à x^i
  - d'un modèle génératif, de paramètres θ
- On veut trouver les paramètres du modèle qui expliquent au mieux la génération des données
- On se donne un critère
  - Ici on considère la vraisemblance des données qui est le critère le plus fréquent
    - $P(D; \theta) = P(x^1, \dots, x^N; \theta)$
  - D'autres critères sont également utilisés
- On va essayer de déterminer les paramètres θ de façon à maximiser la vraisemblance

EM Introduction
Exemple : mélange de deux populations

- On recueille des données sur deux populations
  - e.g. taille d'individus $D = \{x^i; i = 1 \dots N\}$
  - Hypothèse : les données de chaque population sont gaussiennes à 1 dimension
    - $N(\mu_1, \sigma_1), N(\mu_2, \sigma_2)$
- Problème
  - Estimer les μ_i et les σ_i à partir des données
  - Si les d^i sont connus, i.e. $D = \{(x^i, d^i); i = 1 \dots N\}$ la solution est simple
    - On a deux population séparées (2 classes) $C1, C2$
    - On utilise les estimateurs classiques de la moyenne et de la variance
      - Pour la moyenne $\mu_j = \frac{1}{|C_j|} \sum_{x^i \in C_j} x^i$, idem pour la variance
    - Ces estimateurs usuels sont les estimateurs du maximum de vraisemblance
      - cf. slide suivant

EM Introduction - Mélange de deux populations
Cas où l'appartenance est connue

- Vraisemblance
  - $P(D|\theta) = \prod_{x^i \in C_1} p(x^i|\ \theta_1) \prod_{x^j \in C_2} p(x^j|\ \theta_2)$
- En pratique on maximise la log-vraisemblance
  - $L(\theta) = \log\big(P(D|\theta)\big) = \sum_{x^i \in C_1} logp\big(x^i|\theta_1\big) + \sum_{x^j \in C_2} logp\big(x^j|\theta_2\big)$
  - Cas des gaussiennes
    - $p(x|C_k) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\mu_k)^2}{2\sigma_k^2})$
    - $\frac{\partial L}{\partial \mu_k} = 0 \Leftrightarrow \mu_k = \frac{1}{|C_k|} \sum_{x^i \in C_k} x^i$, idem pour la variance

EM Introduction - Mélange de deux populations
Cas où la probabilité d'appartenance est connue

- On connait les $p(C_k|x)$, k = 1, 2 pour tous les x de l'ensemble d'apprentissage
- $p(x^i|\theta) = p(x^i|C_1)p(C_1) + p(x^i|C_2)p(C_2)$
- C'est un modèle de mélange de deux densités

- Log-vraisemblance
  - $L(\theta) = \log(P(D|\theta)) = \sum_{x^i} \log(p(x^i|C_1)p(C_1) + p(x^i|C_2)p(C_2))$
  - Cas des gaussiennes
    - $\frac{\partial L}{\partial \mu_k} = 0 \Leftrightarrow \mu_k = \frac{\sum_{x^i} p(C_1|x^i).x^i}{\sum_{x^i} p(C_1|x^i)}$, idem pour la variance

EM Introduction - Mélange de densités gaussiennes
La probabilité d'appartenance est inconnue

- On suppose que les données sont générées par le modèle suivant
  - Tirer $z^i \sim Multinomiale\ (\phi)\ \phi_i \geq 0, \sum_{j=1}^{k} \phi_i = 1$
  - Tirer $x^i \quad / \quad x^i | z^i = j \sim N(\mu_j, \sigma_j)$
  - i.e. x^i est généré en choisissant d'abord $z^i \in \{1, \dots, k\}$ et ensuite en effectuant un tirage suivant une gaussienne $N(\mu_j, \sigma_j)$ si $z^i = j$
- C'est ce qu'on appelle un mélange de gaussienne
- la vraisemblance des données s'écrit
  - $l(\theta) = l(\mu, \sigma, \phi) = \prod_{i=1}^{N} p(x^i; \theta) = \prod_{i=1}^{N} \sum_{j=1}^{k} p(x^i, z^i = j; \theta)$

$$= \prod_{i=1}^{N} \sum_{j=1}^{k} p(x^i | z^i = j; \mu, \sigma) p(z^i | \phi)$$

- La log vraisemblance
  - $l(\mu, \sigma, \phi) = \sum_{i=1}^{N} \log(\sum_{j=1}^{k} p(x^i | z^i = j; \mu, \sigma) p(z^i | \phi))$
- Les deux exemples précédents (appartenance connue et probabilité d'appartenance connue sont des cas particuliers de ce modèle de mélange)
- Quand la probabilité d'appartenance est inconnue, on ne peut pas trouver une forme analytique pour les estimateurs du maximum de vraisemblance

Algorithme E.M. – Cas général

- Problème
  - On a un problème d'estimation pour lequel on connait un ensemble d'apprentissage $D = \{x^1, \dots, x^N\}$
  - On veut trouver les paramètres qui maximisent la (log) vraisemblance des données $\mathrm{L}(\theta) = \log p(D; \theta)$
  - On ne connait pas de formule analytique permettant d'estimer les paramètres θ
- On postule
  - l'existence de variables cachées z responsables de la génération des données
    - À chaque x^i, on associe sa classe cachée z^i

$$Z = \{z^i; i = 1..N\}$$

  - l'existence d'une fonction densité jointe sur les données observées et cachées $p(x, z)$
    - $\mathrm{p}(D, Z|\theta)$ sera appelé **vraisemblance complète** des données pour le modèle θ.
- Remarque
  - Les variables z sont inconnues et sont considérées comme des variables aléatoires
  - $P(D, Z|\theta)$ sera donc elle même une variable aléatoire

Algorithme EM - Cas général

- L'algorithme E.M. fournit une solution à notre problème
- C'est un algorithme itératif en 2 étapes
  - Etape E. (Espérance)
    - Construire une borne inférieure de $L(\theta)$
    - Cette borne inférieure est **l'espérance** de la vraisemblance complète des données
  - Etape M. (Maximisation)
    - Déterminer les paramètres , i.e. les $\phi_j, \mu_j, \sigma_j, j = 1, \dots, k$ qui optimisent cette borne inférieure

Algorithme EM – Cas general

- La borne inférieure est définie par une fonction auxiliaire Q
  - Q est l'espérance de la log-vraisemblance des données complètes $\log(p(D, Z; \theta))$, connaissant le modèle courant à l'étape t , $\theta(t)$
  - L'espérance est calculée par rapport à la distribution des variables cachées z connaissant le modèle courant $\theta(t)$: $p(z|x; \theta(t))$
  - $Q(\theta, \theta(t)) = E_Z[\log p(D, Z; \theta); \theta(t)) = \sum_Z \log p(D, Z; \theta) p(Z|D; \theta(t))$

- Dans cette expression :
  - D et $\theta(t)$ sont des constantes
  - z est une variable aléatoire de densité $p(z|x; \theta(t))$, Z est l'ensemble des variables z
  - θ représente les paramètres que l'on veut estimer

Algorithme EM – Cas general

- On va montrer que $Q(\theta, \theta(t))$ est une borne inférieure de $L(\theta)$
- Pour cela on utilisera **l'inégalité de Jensen**
- Théorème
  - Soit f une fonction convexe définie sur R et x une variable aléatoire réelle, alors
  - $E[f(x)] \geq f(E[x])$
  - Si f est strictement convexe $E[f(x)] = f(E[x])$ si et seulement si x est une constante
- Rq
  - f est convexe si $f''(x) \geq 0 \; \forall x$
  - Pour une fonction concave (e.g. $\log()$), l'inégalité est vérifiée dans le sens opposé $E[f(x)] \leq f(E[x])$

Algorithme EM – Cas general

- $\log p(D;\theta) = \sum_{i=1}^{N} \log p\left(x^i;\theta\right)$
- $= \sum_{i=1}^{N} \log \sum_{z^i} p\left(x^i, z^i;\theta\right)$
- $= \sum_{i=1}^{N} \log \sum_{z^i} \frac{p\left(z^i|x^i;\theta(t)\right)p\left(x^i,z^i;\theta\right)}{p\left(z^i|x^i;\theta(t)\right)}$
- $= \sum_{i=1}^{N} \log E_{z^i \sim p(z|x;\theta(t))} \frac{p\left(x^i,z^i;\theta\right)}{p\left(z^i|x^i;\theta(t)\right)}$
- $\geq \sum_{i=1}^{N} E_{z^i \sim p(z|x;\theta(t))} \log \frac{p\left(x^i,z^i;\theta\right)}{p\left(z^i|x^i;\theta(t)\right)}$
- $= \sum_{i=1}^{N} \sum_{z^i} p\left(z^i|x^i;\theta(t)\right) log \frac{p\left(x^i,z^i;\theta\right)}{p\left(z^i|x^i;\theta(t)\right)}$
- On a utilisé la concavité de la fonction log pour l'inégalité
- Comme dans l'étape M, on maximise par rapport à θ, on peut éliminer le dénominateur du log dans l'expression précédente, on a alors la fonction auxiliaire :
- $$Q\left(\theta,\theta(t)\right) = \sum_{i=1}^{N} \sum_{z^i} p\left(z^i|x^i;\theta(t)\right) \log p\left(x^i,z^i;\theta\right)$$
- En maximisant Q, on maximise une borne inférieure de la vraisemblance

Algorithme EM – Cas general

- On peut montrer ensuite la convergence de l'algorithme par :
  - $Q(\theta(t+1), \theta(t)) \geq Q(\theta(t), \theta(t)) \Rightarrow p(D; \theta(t+1) \geq p(D; \theta(t))$

Algorithme EM - Cas général

Initialiser $\theta(0)$

1. Etape E : *Espérance*

On calcule $p(Z|D, \theta(t))$

On en déduit $Q(\theta; \theta(t))$

L'espérance est calculée par rapport à la distribution de Z pour les paramètres courants $\theta(t)$

2. Etape M : *Maximisation*

Etant donnée la distribution courante sur Z, trouver les paramètres qui maximisent Q

$$\theta(t+1) = argmax_\theta E_{z \sim p(z|x;\theta(t))}[\log p(D, Z; \theta)]$$

- L'algorithme converge vers un maximum local de la fonction Q et de $p(D; \theta)$

Algorithme EM - Cas général

- Remarques
  - L'algorithme est utilisé pour
    - les algorithmes non supervisés, semi - supervisés
    - les données manquantes ou les composantes manquantes dans les données
    - les HMM ...

Algorithme EM - Mélange de densités – cas gaussien

- On suppose que le modèle génératif des données est un mélange de densités gaussiennes
  - On fixe a priori le nombre de composantes du mélange à k
  - Pour simplifier, on suppose que les données x sont unidimensionnelles
  - $p(x) = \sum_{l=1}^{k} p(l)p(x|l), \qquad p(x|l) = \frac{1}{(2\pi\sigma_l^2)^{\frac{1}{2}}} \exp(-\frac{(x-\mu_l)^2}{2\sigma_l^2})$

- Paramètres
  - Coefficients du mélange $\phi_l = p(l)$, moyennes μ_l et écarts types σ_l :

$$\theta = \{p(l), \mu_l, \sigma_l; l = 1 \dots k\}$$

Algorithme EM - Mélange de densités – cas gaussien

- Log-vraisemblance
  - $\log p(D;\theta) = \sum_{i=1}^{N} \log \sum_{l=1}^{k} p(l;\theta) p(x^i|l;\theta)$
- Vraisemblance complète
  - $\log p(D,Z;\theta) = \sum_{i=1}^{N} \log(p(z^i;\theta)p(x^i|z^i;\theta)) = \sum_{i=1}^{N}\sum_{l=1}^{k} \delta_{lz^i} \log(p(l;\theta)p(x^i|l;\theta))$
- Fonction auxiliaire
  - $Q(\theta,\theta(t)) = \mathrm{E}_{z\sim p(z|x;\theta(t))}[\log p(D,Z;\theta)]$
  - $= \sum_{z^1=1}^{k} \dots \sum_{z^N=1}^{k} \log p(D,Z;\theta) \prod_{i=1}^{N} p(z^i|x^i;\theta(t))$
  - quelques réécritures …(cf slide suivant) conduisent à l'expression :
  - $= \sum_{i=1}^{N}\sum_{l=1}^{k} p(l|x^i;\theta(t)) \log(p(l;\theta)p(x^i|l;\theta))$
  - Qui sera utilisée dans la maximisation de l'étape M.

- Reécriture de la fonction auxiliaire
  - $Q = Q\left(\theta, \theta(t)\right) = \mathrm{E}_{\mathbf{z}\sim \mathrm{p}(\mathbf{z}|\mathrm{x};\theta(t))}[\log \mathrm{p}(\mathrm{D}, \mathrm{Z}; \theta)]$
  - $Q = \sum_{z^1=1}^{k} \dots \sum_{z^N=1}^{k} \log p(D, Z; \theta) \prod_{i=1}^{N} p(z^i|x^i; \theta(t))$
  - $Q = \sum_{z^1=1}^{k} \dots \sum_{z^N=1}^{k} (\sum_{\mathrm{j}=1}^{N} \sum_{l=1}^{k} \delta_{lz^j} \log(p(l; \theta) p\left(x^j \middle| l; \theta\right))) \prod_{i=1}^{N} p(z^i|x^i; \theta(t))$
  - $Q = \sum_{j=1}^{N} \sum_{l=1}^{k} [\sum_{z^1=1}^{k} \dots \sum_{z^N=1}^{k} \delta_{lz^j} \prod_{i=1}^{N} p\left(z^i \middle| x^i; \theta(t)\right)] \log p(l; \theta) p\left(x^j \middle| l; \theta\right)]$
  - $A = \sum_{z^1=1}^{k} \dots \sum_{z^N=1}^{k} \delta_{lz^j} \prod_{i=1}^{N} p\left(z^i \middle| x^i; \theta(t)\right)]$
  - $A = \sum_{z^1=1}^{k} \dots \sum_{z^{j'-1}=1}^{k} \sum_{z^{j'+1}=1}^{k} \dots \sum_{z^N=1}^{k} (\sum_{z^{j'}=1}^{k} \delta_{lz^{j'}} \prod_{i=1}^{N} p\left(z^i \middle| x^i; \theta(t)\right))$
  - $A = \sum_{z^1=1}^{k} \dots \sum_{z^{j'-1}=1}^{k} \sum_{z^{j'+1}=1}^{k} \dots \sum_{z^N=1}^{k} (p(l|x^{j'}; \theta(t)) \prod_{i\neq j'} p\left(z^i \middle| x^i; \theta(t)\right))$
  - $\sum_{z^1=1}^{k} \dots \sum_{z^{j'-1}=1}^{k} \sum_{z^{j'+1}=1}^{k} \dots \sum_{z^N=1}^{k} \prod_{i\neq j'} p\left(z^i \middle| x^i; \theta(t)\right)) = 1$
  - $Q = \sum_{j=1}^{N} \sum_{l=1}^{k} p(l|x^j; \theta(t)) \log p(l; \theta) p\left(x^j \middle| l; \theta\right)$

Algorithme EM - Mélange de densités – cas gaussien – Etapes E et M

- Etape E
  - $p\left(z^i = j \middle| x^i; \theta(t)\right) = \frac{p\left(x^i \middle| z^i = j; \theta(t)\right) p\left(z^i = j; \theta(t)\right)}{\sum_{l=1}^{k} p\left(x^i \middle| z^i = l; \theta(t)\right) p\left(z^i = l; \theta(t)\right)} \quad \forall i, j$
- Etape M
  - $Min_\theta(-Q\left(\theta, \theta(t)\right)$ sous la contrainte $\sum_{l=1}^{k} p(l; \theta) = 1$
  - Équivalent à
  - $Min_\theta(-Q\left(\theta, \theta(t)\right) + \lambda(\sum_{l=1}^{k} p(l; \theta) - 1)$
    - Avec λ le coefficient de Lagrange associé

- Remarque
  - Dans l'étape E on n'a pas besoin de calculer explicitement $Q\left(\theta, \theta(t)\right)$, il suffit de calculer les $p\left(z^i = j \middle| x^i; \theta(t)\right)$ (cf dérivation du résultat en cours)

Algorithme EM - Mélange de densités – cas gaussien – Reestimation dans l'étape M

- à l'étape M à l'itération *t*, on obtient les formules de reestimation suivantes
  - $p(j) = \frac{1}{N}\sum_{i=1}^{N} p\left(z^i = j \middle| x^i; \theta(t)\right)$
  - $\mu_j = \frac{\sum_{i=1}^{N} p\left(z^i = j \middle| x^i; \theta(t)\right) \mathrm{x^i}}{\sum_{i=1}^{N} p\left(z^i = j \middle| x^i\right)}$
  - $\sigma_j^2 = \frac{\sum_{i=1}^{N} p\left(z^i = j \middle| x^i; \theta(t)\right)\left(x^i - \mu_j\right)^2}{\sum_{i=1}^{N} p\left(z^i = j \middle| x^i\right)}$

Algorithme EM - Mélange de densités – cas gaussien

- Etape E
  - Si on connait les lois de mélange (les (μ, σ, ϕ)), on peut facilement calculer les $p(z^i = j|x^i)$ par $p(z^i = j|x^i) = \frac{p(x^i|z^i=j)p(z^i=j)}{p(x^i)}$
- Etape M
  - Si on connait les probabilités d'appartenance, on peut facilement estimer les paramètres des lois de mélange comme vu dans l'exemple

- Il suffit de partir d'une valeur initiale – pour les $p(z^i = j|x^i)$ ou pour les paramètres des lois de mélange pour exécuter l'algorithme
  - Celui ci converge vers un maximum local de la vraisemblance

Apprentissage non supervisé

Mélange de densités

Apprentissage par échantillonnage de Gibbs

Les méthodes MCMC
Markov Chain Monte Carlo

- Méthodes de calcul intensif basées sur la simulation pour
  - Echantillonnage de variables aléatoires
    - $\{x^t\}_{t=1..T}$ qui suivent une certaine distribution $p(x)$
  - Calcul de l'espérance de fonctions suivant cette distribution
    - $E[f(x)]$ sera estimé par $\frac{1}{T}\sum_{t=1}^{T} f(x^t)$
    - e.g. moyenne, marginales, …
  - Maximisation de fonctions
    - $Argmax_x p(x)$

Echantillonneur de Gibbs

- On veut estimer une densité $p(\boldsymbol{x}), \boldsymbol{x} \in R^n$, avec $\boldsymbol{x} = (x_1, \dots, x_n)$
- Hypothèse
  - On connait les lois conditionnelles
    - $p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ qui sera notée $p(x_i|x_{-i})$
- Algorithme
  - Initialiser $\{x_i^0, i = 1, \dots, n\}$
  - Itérer jusqu'à convergence (variable d'itération notée t)
  - échantilloner
    - $x_1^{t+1} \sim p(x_1|x_{-1}^t) = p(x_1|x_2^t, \dots, x_n^t)$
    - …..
    - $x_n^{t+1} \sim p(x_n|x_{-n}^t) = p(x_n|x_1^{t+1}, \dots, x_{n-1}^{t+1})$
  - Jusqu'à ce que la distribution jointe $p(x_1, \dots, x_n)$ ne change plus

- Propriétés
  - Sous certaines conditions de régularité, la procédure converge vers la distribution cible $p(\boldsymbol{x})$
    - Les échantillons résultants sont des échantillons de la loi jointe $p(\boldsymbol{x})$
  - On n'a pas besoin de connaitre la forme analytique des $p(x_i|x_{-i})$ mais uniquement de pouvoir échantillonner à partir de ces distributions
    - Mais la forme analytique permet d'avoir de meilleurs estimés
  - Avant de retenir les points échantillons, on autorise souvent une période de "burn-in" pendant laquelle on fait simplement tourner l'algorithme "à vide"
  - Gibbs facile à implémenter, adapté aux modèles hierarchiques (cf LDA)

Cas du mélange de deux lois gaussiennes

- Modèle : $p(x) = \sum_{l=1}^{2} p_l p(x|l)$
  - On considère un mélange de deux gaussiennes à une dimension
- On va considérer un modèle augmenté en ajoutant une variable cachée $z \in \{0,1\}$
  - Les données complètes sont les (x^i, z^i)
- Les paramètres à estimer sont
  - Les paramètres des gaussiennes : $\theta = \{p_l, \mu_l, \sigma_l; l = 1,2\}$
  - La variable cachée z (qui est considérée comme un paramètre supplémentaire à estimer dans la procédure de gibbs)
- On va utiliser Gibbs en échantillonnant sur les densités conditionnelles des paramètres $p(\theta, z|x)$
  - Pour simplifier on suppose dans l'example que les proportions p_l et les variances σ_l sont fixées, on estime juste les moyennes μ_1, μ_2
- Pour cela, on va échantillonner suivant la distribution jointe$p((\mu_1, \mu_2), z|x)$

Echantillonneur de Gibbs pour le modèle de mélange de deux gaussiennes

- Hyp: $p_i$et $\sigma_i, i = 1,2$ sont supposés connus
- Choisir des valeurs initiales $\mu_1(0), \mu_2(0)$
- Répéter ($t = 1,2, \ldots$) jusqu'à convergence (i.e. $\mu_j(t), j = 1,2$ ne changent plus)
  - Pour $i = 1 \ldots N$
    - Générer $z^i(t) \in \{0,1\}$ selon la distribution
      - $p(z^i = 1|x^i) = \frac{p_1 p(x^i|z^i=1;\mu_1(t-1),\sigma_1)}{p_1 p(x^i|z^i=1;\mu_1(t-1),\sigma_1)+p_2 p(x^i|z^i=0;\mu_2(t-1),\sigma_2)}$
      - Rq: c'est une loi de Bernouilli de paramètre le membre droit de l'équation
  - Calculer
    - $\hat{\mu}_1(t) = \frac{\sum_{i=1}^N z^i(t)x^i}{\sum_{i=1}^N z^i(t)}$
    - $\hat{\mu}_2(t) = \frac{\sum_{i=1}^N (1-z^i(t))x^i}{\sum_{i=1}^N (1-z^i(t))}$

    Et générer $\mu_j(t) \sim N(\hat{\mu}_j, \sigma_j), j = 1,2$

Lien avec l'algorithme EM

- Les étapes pour cet exemple sont les mêmes qu'avec EM
- Différence
  - Au lieu de maximiser la vraisemblance, aux étapes 1 et 2, on échantillonne
    - Etape 1 : on simule les variables cachées z à partir de la distribution conditionnelle $p(z|\theta, x)$ au lieu de calculer $E(z|\theta, x)$ dans EM
    - Etape 2 : on simule à partir de $p(\mu 1, \mu 2\ |z, x)$ au lieu de calculer le max. vraisemblance $p(\mu 1, \mu 2\ , z|x)$ dans EM

- Remarque 1
  - Pour simplifier on a supposé que les proportions p_i et les variances $\sigma_i, i = 1,2$ étaient fixes.
  - Dans le cas général où elles ne sont pas connues, on va
    - 1. fixer des valeurs initiales de l'ensemble des paramètres $\theta = \{p_i, \mu_i, \sigma_i; i = 1,2\}$ à partir de distribution a priori pour ces paramètres
    - 2. échantillonner alternativement pour chaque paramètre selon sa distribution a posteriori, conditionnellement aux autres paramètres
      - Par exemple on échantillonnera successivement pour (p_1, p_2), (μ_1, μ_2), puis (σ_1, σ_2)

- Remarque 2
  - Il existe d'autres méthodes pour estimer des densités de façon approchée quand il n'existe pas de forme analytique conduisant à des calculs, e.g. méthodes variationnelles

Apprentissage non supervisé

Spectral Clustering

Spectral Clustering (after Von Luxburg 2007)

- Intuition
  - $\mathbf{x}_1, \ldots, \mathbf{x}_n$ data points, w_{ij} similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$
  - G = (V, E) undirected graph
    - $V = \{v_1, \ldots, v_n)$, vertex v_i corresponds to data point $\mathbf{x}_i$
    - Edges are weighted, $W = (w_{ij})_{i,j = 1 \ldots n}$, $w_{ij} \geq 0$ is the weight matrix
    - D : diagonal matrix with $d_i = \sum_{j=1}^{n} w_{ij}$
  - Clustering amounts at finding a graph partition such that
    - Edges between clusters have low weights
    - Edges among points inside a cluster have high values

Spectral Clustering

- Building similarity graphs from data points
  - Different ways to build a similarity graph
  - Build a locally connected graphs: k-nearest neighbor graphs
    - Two vertices are connected if one of them is among the k-nearest neighbor of the other
    - Or two vertices are connected if both are in the k-neighborhood of the other
    - Edges are then weighted using the similarity of the vertices
  - Build a fully connected graphs
    - $w_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2)$

Spectral Clustering

- Graph Laplacians
  - Unnormalized graph Laplacian
    - $L = D - W$
  - Normalized graph Laplacians
    - $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = \mathrm{I} - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ symmetric
    - $L_{rw} = D^{-1} L = I - D^{-1} W$ interpretation : random walk on the graph

Spectral Clustering

- Properties of the unnormalized graph Laplacian L
- Proposition 1- L satisfies:
  - $\forall\, y \in R^n, \quad y^T L y = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(y_i - y_j\right)^2$
  - L is symmetric, positive semi-definite
  - The smallest eigenvalue of L is 0, the corresponding eigenvector is $\mathbf{1}$ (vector with n $1s$)
  - L has n non negative eigenvalues $0 = \lambda_1 \leq \ \ldots \leq \lambda_n$
- Proof
  - $y^T L y = y^T D y - y^T W y = \sum_{i=1}^{n} d_i y_i^2 - \sum_{i,j=1}^{n} y_i y_j w_{ij} = \frac{1}{2}\left(\sum_{i=1}^{n} d_i y_i^2 - 2\sum_{i,j=1}^{n} y_i y_j w_{ij} + \sum_{j=1}^{n} d_j y_i^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(y_i - y_j\right)^2$
  - Symmetry follows from the symmetry of L and W, positive semi definiteness comes from $y^T L y \geq 0$
  - Obvious: a symmetric, positive semi-definite matrix has all its eigenvalues ≥ 0
  - Direct consequence of properties 1 and 3

Spectral Clustering

- Proposition 2 (number of connected components and spectrum of L)
  - Let G be an undirected graph with non negative weights. The multiplicity *k* of the eigenvalue 0 of *L* equals the number of connected components in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors of these components.
  - This result provides intuition on spectral clustering algorithms
  - Connected graph
    - Let us consider first $k = 1$. If y is an eigenvector with eigenvalue 0 then $0 = y^T L y = \sum_{i=1}^{n} w_{ij}(y_i - y_j)^2$. Since $w_{ij} \geq 0$ this is only possible if $y_i = y_j \ \forall i, j$. In a graph with 1 connected component, 1 the constant vector filled with $1s$ is the eigenvector with eigenvalue 0. It is also the indicator vector of the connected component
  - Multiple components
    - If there are k connected components. The W matrix and hence L can be arranged in a block diagonal form. The above result holds for each connected component A_i. The corresponding eigenvectors are the 1_{A_i} with $1s$ corresponding to the i^{th} block and $0s$ everywhere else.
    - If one can identify these eigenvectors, one will identify the clusters
      - This is only intuition, the situation is usually more complex (see Luxburg 2007)

Spectral Clustering

- Properties of the normalized graph Laplacians
  - $\forall\, y \in R^n, \quad y^T L_{sym} y = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{y_i}{\sqrt{d_i}} - \frac{y_j}{\sqrt{d_j}}\right)^2$
  - L_{sym} and L_{rw} are positive semi-definite and have n non negative eigenvalues $0 = \lambda_1 \leq \ \ldots \leq \lambda_n$
  - λ is an eigenvalue of L_{rw} with eigenvector u iff λ is an eigenvalue of L_{sym} with eigenvector $D^{1/2}u$

Unnormalized spectral clustering algorithm

- Idea
  - Project data points $\mathrm{x_i} \in R^m, \mathrm{i} = 1 \ldots \mathrm{n},$ in a smaller dimensional space, say of dimension k where clustering is performed
- Input: n points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, similarity matrix S
- Output: clusters
  - Construct similarity graph and corresponding weight matrix W
  - Compute unnormalized Laplacian L
  - Compute k first eigenvectors of L (corresponding to smallest eigenvalues): $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$
  - U: $n\ x\ k$ matrix with columns $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$
  - For $i = 1 \ldots n, \boldsymbol{y}_i \in R^k$ i-th row of U
  - Cluster the $\boldsymbol{y}_i, i = 1 \ldots n$ with k-means algorithm into clusters $C_1, \ldots, C_k$
- k clusters in the initial space: $C'1, \ldots, C'_k \,/\, C'_i = \{\boldsymbol{x}_j \,/\, \boldsymbol{y}_j \in C_i\}$

- Note: Similar algorithms with normalized Laplacians

Apprentissage non supervisé

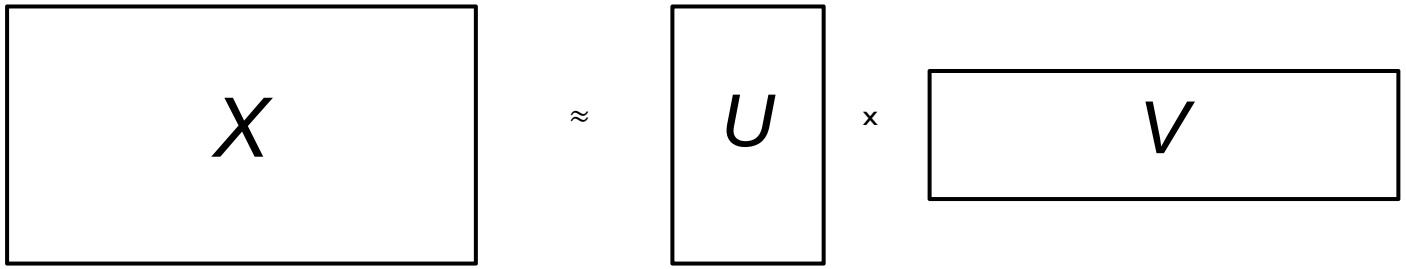
Non Negative Matrix Factorization

Matrix Factorization

- Idea
  - Project data vectors in a latent space of dimension k < m size of the original space
  - Axis in this latent space represent a new basis for data representation
  - Each original data vector will be approximated as a linear combination of k basis vectors in this new space
  - Data are assigned to the nearest axis
  - This provide a clustering of the data

Matrix factorization

- $X = \{x_{.1}, \ldots, x_{.n}\}, x_i \in R^m$
- X m x n matrix with columns the $x_{.i}$ s
- Low rank approximation of X
  - Find factors $U,V, / X \approx UV$
  - With U an m x k matrix, U a k x n matrix, k < m, n



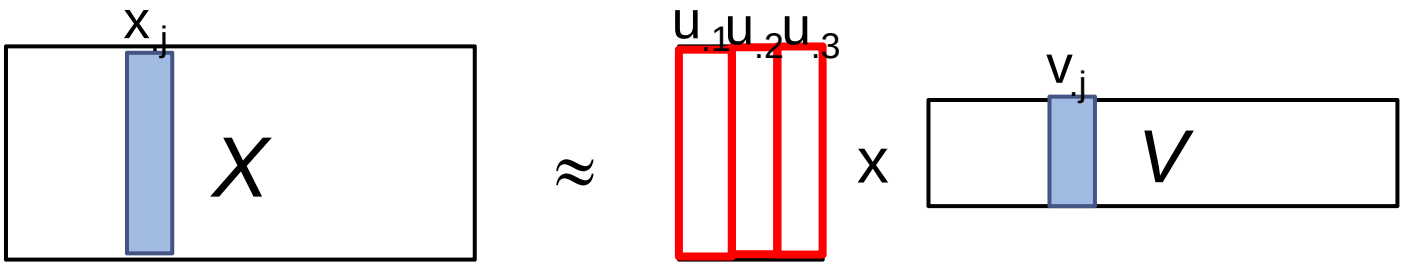
- Many different decompositions
  - e.g. Singular Value Decomposition, Non Negative Matrix Factorization, Tri factorization, etc

- Applications
  - Recommendation (User x Item matrix)
    - Matrix completion

	Star wars	Terminator	Titanic	Scarface
Bob	5	4	3	?
Alice	4	?	5	3
John	?	5	3	4

  - Link prediction (Adjacency matrix)
  - …

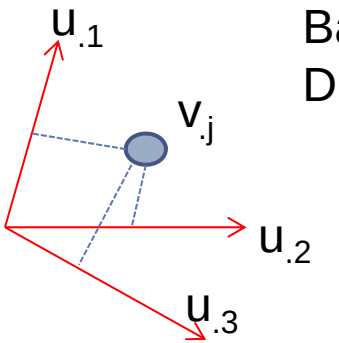
- $X \approx UV$
  - $x_{.j} = \sum_{j=1}^{k} u_{.i} v_{ij}$
  - Columns of U, $u_{.j}$ are basis vectors, the v_{ij} are the coefficient of $x_{.j}$ in this basis



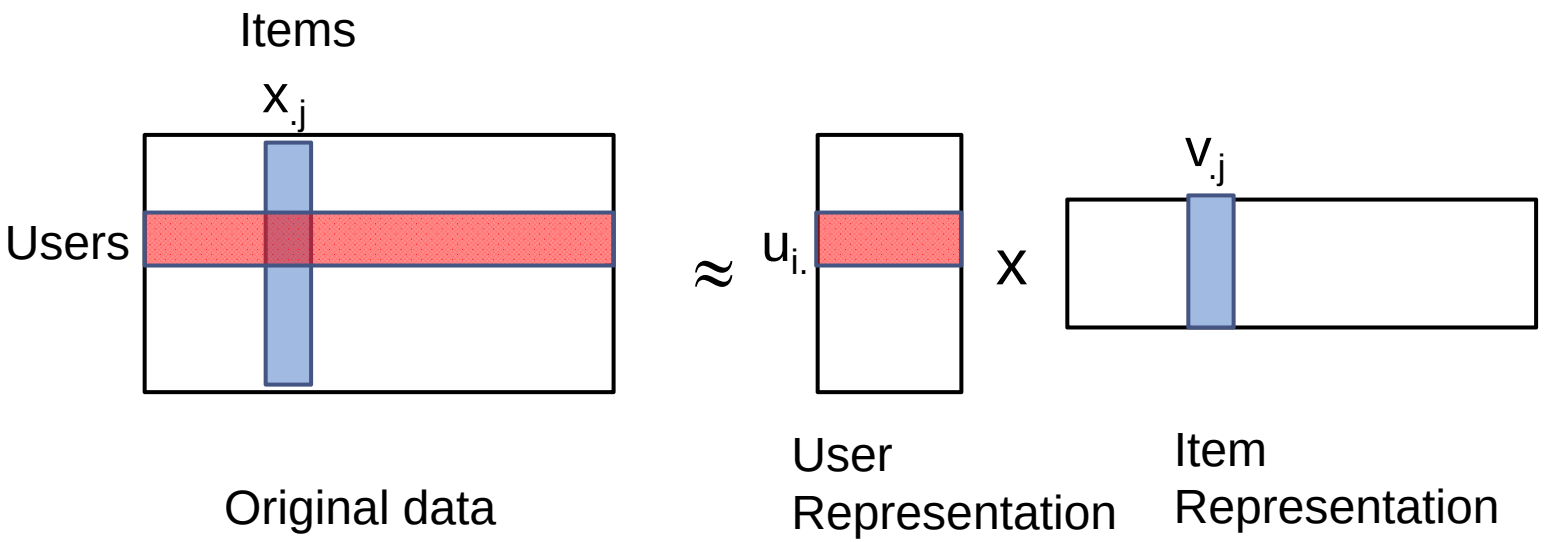
Original data

Basis vectors Dictionnary

Representation

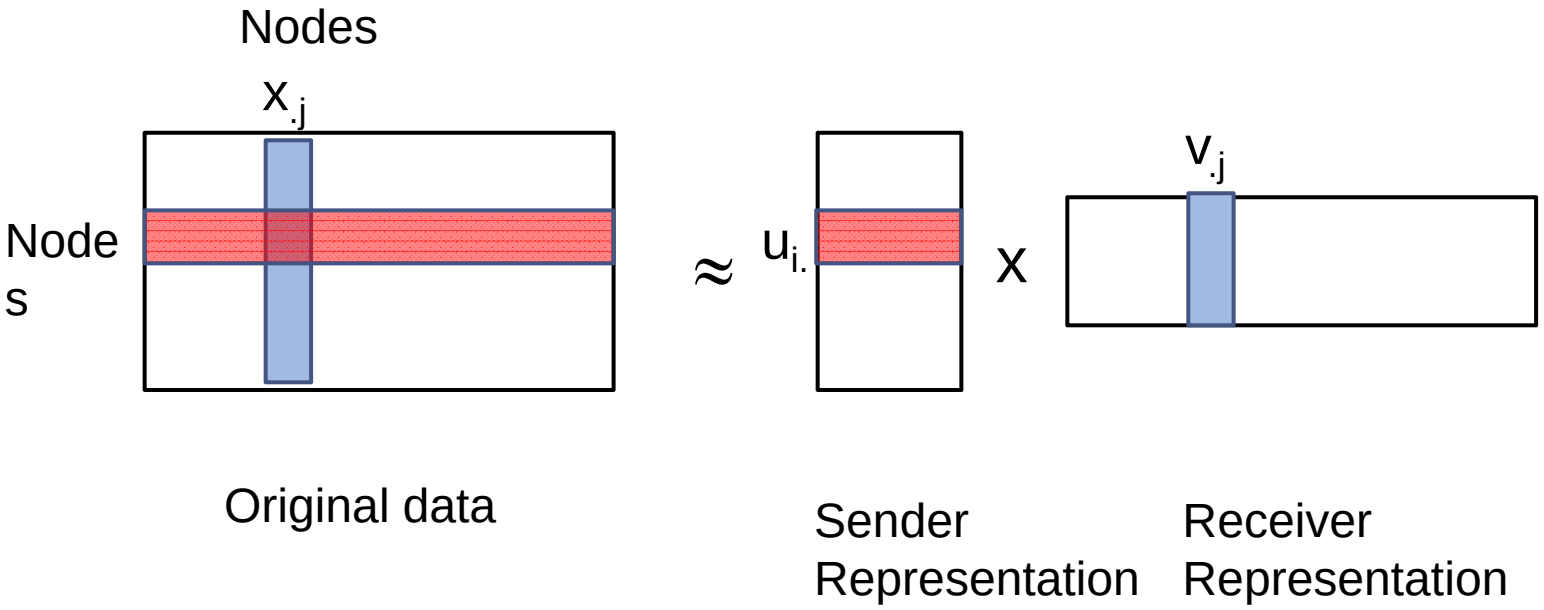


- Interpretation
  - If X is a User x Item matrix



  - Users and items are represented in a common representation space of size k
  - Their interaction is measured by a dot product in this space

- Interpretation
  - If X is a directed graph adjacency or weight matrix



- Loss function - example
- Minimize $C = \|X - UV\|^2 + c(U,V)$
  - constraints on U,V: $c(U,V)$ e.g.
    - Positivity (NMF)
    - Sparsity of representations, e.g. $\|V\|_1$, group sparsity, …
    - Overcomplete dictionary U: k > m
    - Symmetry, e.g. trifactorisation
    - Bias on U and V
      - e.g. biases of users (low scores) or item (popularity) for recommendation
    - Multiple graphs
    - Any a priori knowledge on U and V

Non Negative Matrix Factorization

- Loss function
  - Solve

    $Min_{\{U,V\}} \|X - UV\|^2$

    Under constraints $U, V \geq 0$

    i.e. the factors are constrained to be non negative
  - Convex loss function in U and in V, but not in both U and V

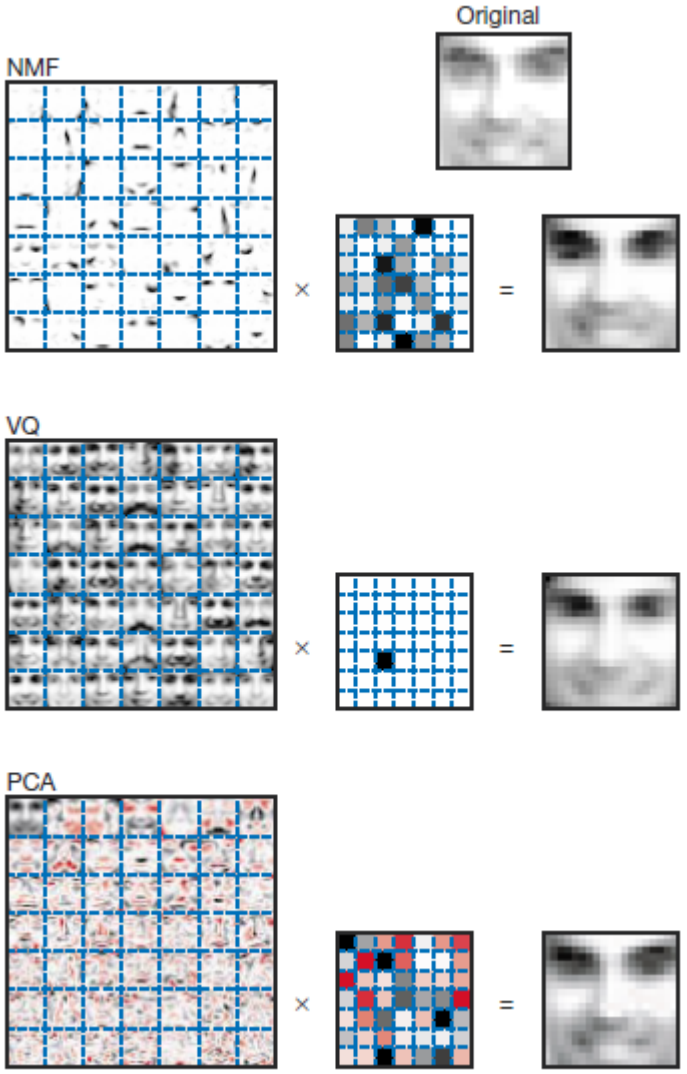
- Algorithm
  - Constrained optimization problem
  - Can be solved by a Lagrangian formulation
    - Iterative multiplicative algorithm (Xu et al. 2003)
      - U,V initialized at random values
      - Iterate until convergence
        - $u_{ij} \leftarrow u_{ij} \frac{(XV^T)_{ij}}{(UVV^T)_{ij}}$
        - $v_{ij} \leftarrow v_{ij} \frac{(X^T U)_{ij}}{(V^T U^T U)_{ij}}$
  - Or by projected gradient formulations
  - The solution *U*, *V* is not unique, if *U*, *V* is solution, then *UD*, $D^{-1}V$ for D diagonal positive is also solution

- Using NMF for Clustering
  - Normalize U as a column stochastic matrix (each column vector is of norm 1)
    - $u_{ij} \leftarrow \frac{u_{ij}}{\sqrt{\sum_i \ u_{ij}^2}}$
    - $v_{ij} \leftarrow v_{ij} \sqrt{\sum_i \ u_{ij}^2}$
  - U columns are cluster centroids, V columns are cluster membership indicators.
  - Under the constraint "U normalized" the solution U, V is unique
  - Associate x^i to cluster j if $j = argmax_j(v_{ji})$

- Note
  - Many different versions and extensions of NMF
  - Different loss functions
    - e.g. different constraints on the decomposition
  - Different algorithms
- Applications
  - Clustering
  - Recommendation
  - Link prediction
  - Etc
- Specific forms of NMF can be shown equivalent to
  - PLSA
  - Spectral clustering

Illustration (Lee & Seung 1999)

- Basis images for
- NMF
- Vector Quantization
- Principal Component Analysis



- Citations
  - Ulrike Luxburg. 2007. A tutorial on spectral clustering. Statistics and Computing 17, 4 (December 2007), 395-416.
  - Daniel D. Lee and H. Sebastian Seung (1999). "Learning the parts of objects by non-negative matrix factorization". Nature 401 (6755): 788–791.
  - W. Xu, X. Liu, and Y. Gong. (2003) Document clustering based on non-negative matrix factorization. SIGIR '03, page 267
  - Yang and E. Oja. Linear and nonlinear projective nonnegative matrix factorization. IEEE Trans. on Neural Networks, 21:734–749, 2010.