INF8225

Leçon 8

Christopher Pal
École Polytechnique de Montréal

Plan du cours

- Discussion de notre contrôle périodique
- Discussion des articles de recherche et projets
- RNNs, LSTMs, Machines de Boltzmann

Une discussion très concise des méthodes classiques d'apprentissage automatique

- L'analyse en composantes principales (ACP) et la relation avec d'autres projections simple de données
- Machines à vecteurs de support
- Arbres de décision

Examen intra

- Pendant notre prochain cours – c.à.d. après la semaine de relâche
- 20% de la note finale
- Tout documentation permises
- Assurez vous que vous avez tous les notes que j'ai présentées sur le tableau noir

Grandes thèmes pour l'intra

- Probabilité de base, calculs et règles de base,
- Inférence : quantités probabilistes d'intérêts
- Apprentissage automatique et concepts de base: utilisation d'ensemble de validation, etc.
- Réseaux bayésiens
- Méthode de moindres carrées
- L'approche d'espérance maximisation « EM »
- Les mélanges de densité, la régression logistique
- Rétropropagation, les perceptrons multicouche et CNNs (concepts de base)
- Pas de: « MDPs », « Deep RL », autoencodeurs, RNNs et d'autres choses discuté aujourd'hui

Présentation des articles de recherche et les projets

- Groupes de 4 max dans les deux cas
- Coordonner les groupes pour les articles de recherche pendant le cours -- c.à.d. maintenant
- Discuter le choix avec Prof. Pal
- Préciser une idée préliminaire pour votre projet dans le Google Doc (pendant la semaine de relâche)

Les projections ou transformations (simple) des données

Chris Pal

Principal Component Analysis (PCA)

Chris Pal

Sujets

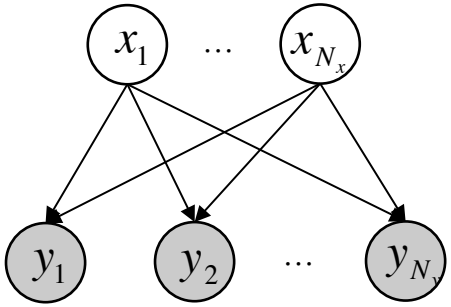
- ACP probabiliste
- Reconstruction
- Exemple: « Eigenfaces »
- ACP & Algèbre linéaire
- Exemple: « Eigendocuments »

Topics

- PCA & Linear Algebra
- Reconstruction
- Eigenfaces & Eigendocuments
- Probabilistic PCA
- Interest Point Descriptors
  (a way to replace the last step of
  Lowe's SIFT descriptor technique)

ACP : l'interprétation probabiliste

- Considérons un modèle gaussien et linéaire avec deux couches, un avec des variables cachées x_i et l'autre avec les observations y_j



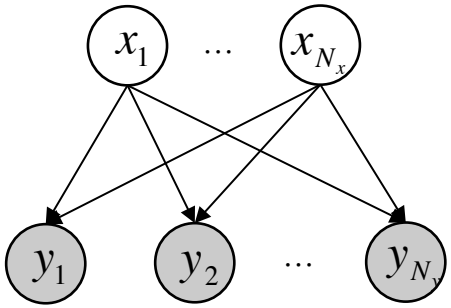
$$p(\mathbf{x}) = N(0,\mathbf{I})$$

$$p(\mathbf{y} \mid \mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{v} \quad \mathbf{v} \sim N(0,\mathbf{R})$$

$$p(\mathbf{y}) = N(\mathbf{0},\mathbf{C}\mathbf{C}^T + \mathbf{R})$$

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} = \frac{N(\mathbf{C}\mathbf{x},\mathbf{R})N(0,\mathbf{I})}{N(\mathbf{0},\mathbf{C}\mathbf{C}^T + \mathbf{R})}$$

ACP : l'interprétation probabiliste



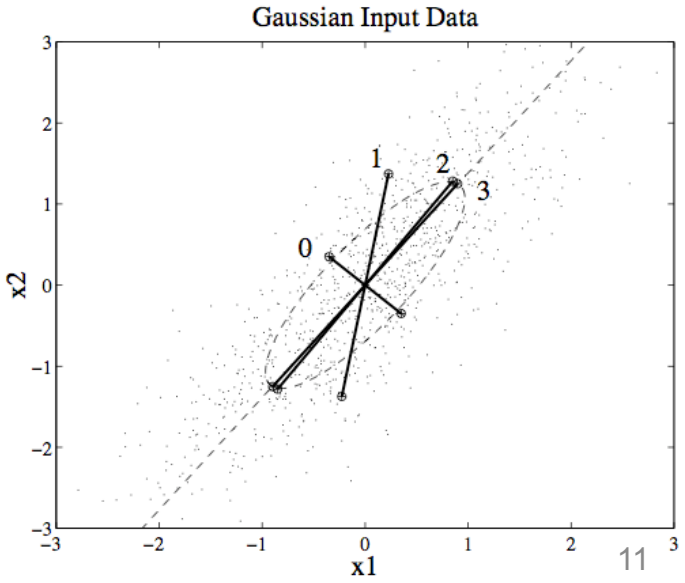
$$p(\mathbf{x} \mid \mathbf{y}) = N(\mathbf{By}, \mathbf{I} - \mathbf{BC}),$$

$$\mathbf{B} = \mathbf{C}^T(\mathbf{CC}^T + \mathbf{R})^{-1}$$

- Dans la limite que le bruit approche zéro on peut obtenir l'approche EM pour ACP comme

Étape E : $\mathbf{X} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{Y}$

Étape M: $\mathbf{C}^{new} = \mathbf{YX}^T(\mathbf{XX}^T)^{-1}$



Les visages d'Olivetti et l'ACP

- Étant donnée des visages et encodant chaque pixel comme un élément dans un vecteur
- « Apprendre » une représentation composé d'un combinaison linéaire capable de reconstruire l'image avec une pondération simple des images, ou des fonctions de base

« Eigenfaces »

mean

principal basis 1

principal basis 2

principal basis 3

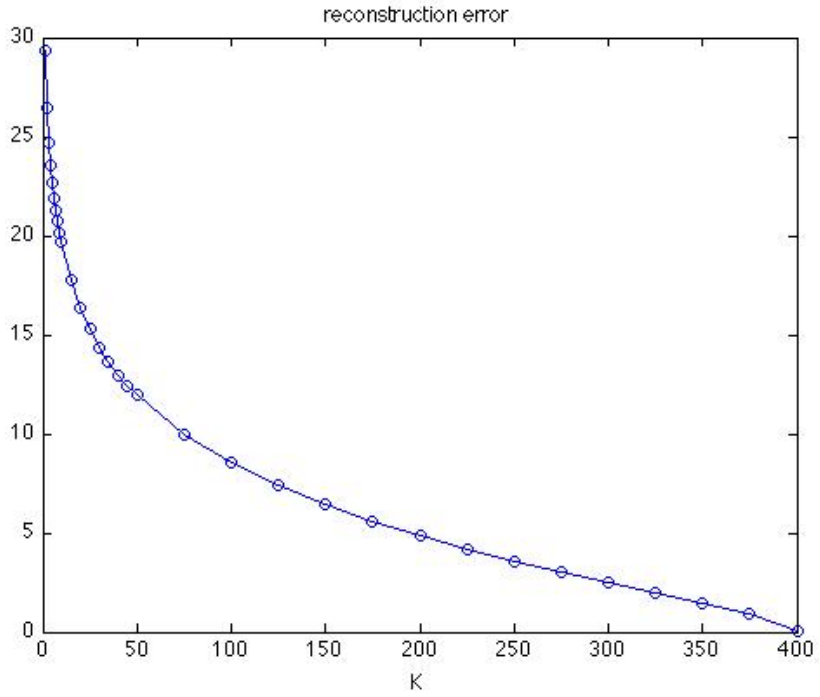
La reconstruction avec les eigen-visages

reconstructed with 2 bases

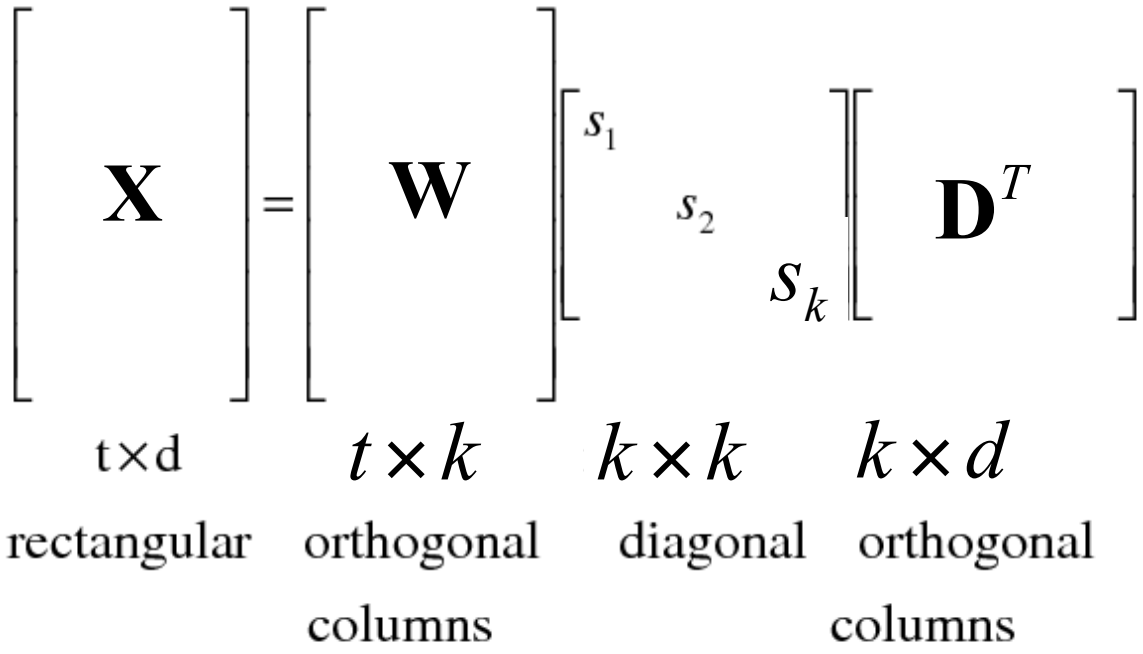
reconstructed with 10 bases

reconstructed with 100 bases

reconstructed with 400 bases

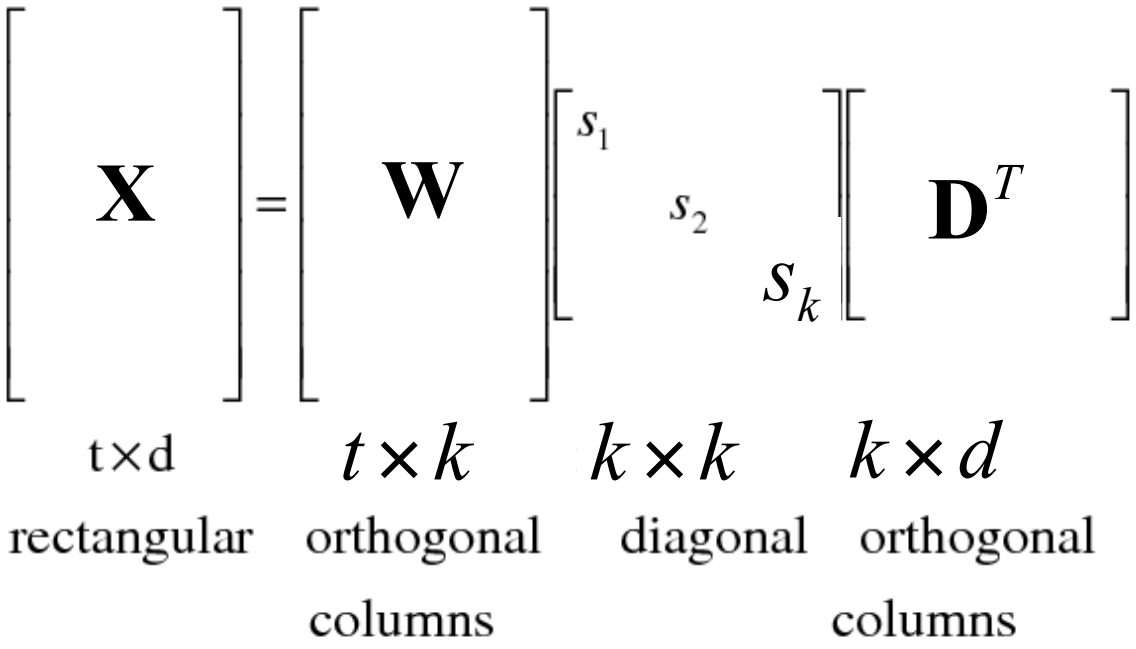


Analyse en composantes principales (ACP) & la décomposition en valeurs singulières (SVD)



- Empilez les données dans les colonnes de la matrice **X**, utilisez un factorisation d'algèbre linéaire (k<<d)
- Décomposez avec **X** = **WSD**$^{\mathbf{T}}$ – (méthode très connu)

Principal Component Analysis (PCA) & the Singular Value Decomposition SVD



- Stack up data as columns of matrix **X**
- Decompose into $\mathbf{X} = \mathbf{WSD}^T$

Reconstructions Approximatif

- Des valeurs singulières sont assorties par taille le long de la diagonale de la matrice **S**
- Avec les *k plus grandes valeurs singulières* il est possible de reconstruire la matrice
- Nous avons $\mathbf{WSD}^T = \widetilde{\mathbf{X}} \approx \mathbf{X}$

$$\underset{t\times d}{\begin{bmatrix} \widetilde{\mathbf{X}} \end{bmatrix}} = \underset{t\times k}{\begin{bmatrix} \mathbf{W} \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_k \end{bmatrix}} \underset{k\times d}{\begin{bmatrix} \mathbf{D}^T \end{bmatrix}}$$

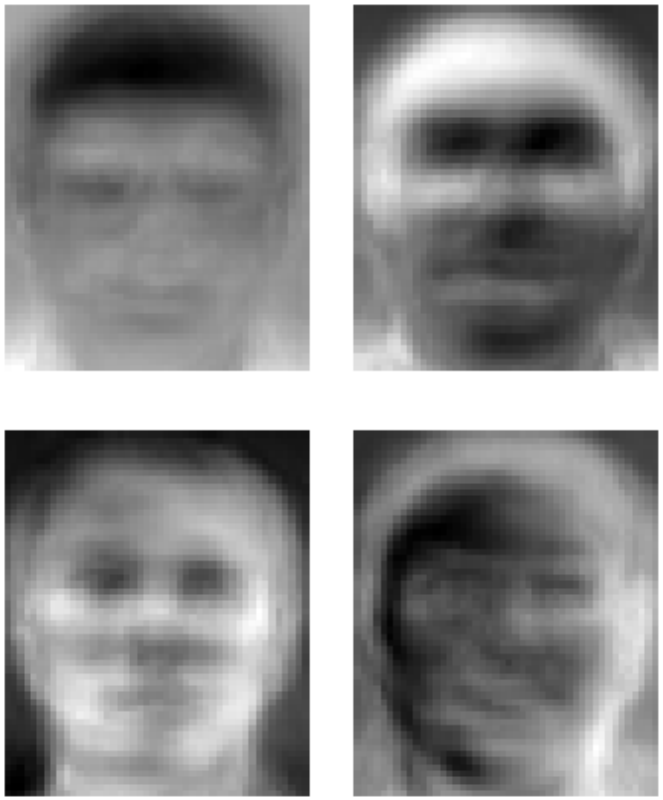
Approximating Data Matrices

- Singular values are sorted by size along the diagonal of matrix **S**
- By keeping the *k largest singular values* it is possible to reconstruct matrix
- We have $\mathbf{WSD}^T = \widetilde{\mathbf{X}} \approx \mathbf{X}$

$$\underset{t\times d}{\begin{bmatrix} \widetilde{\mathbf{X}} \end{bmatrix}} = \underset{t\times k}{\begin{bmatrix} \mathbf{W} \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_k \end{bmatrix}} \underset{k\times d}{\begin{bmatrix} \mathbf{D}^T \end{bmatrix}}$$

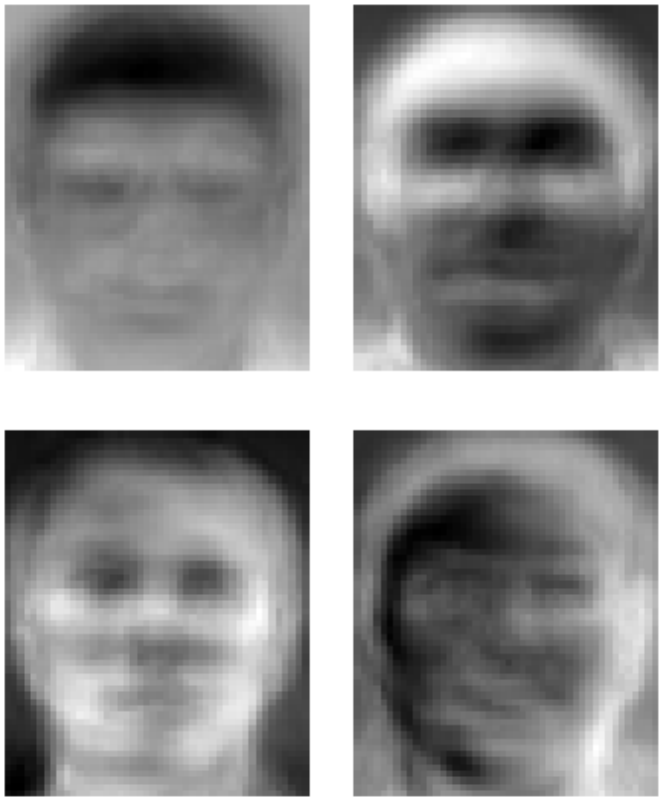
À quoi les vecteurs propres « Eigenvectors » ressemblent-ils ?

- Exemple: « Eigenfaces »
- Appliquez-vous ACP aux images des visages bien alignées
- **W**: Vecteurs propres ou « Eigenvectors » dans les colonnes de la matrice
- **S**: Valeurs propres le long de diagonale de matrice
- Notez les propriétés « globales » de ces vecteurs



What Do Eigenvectors Look Like?

- Example: Eigenfaces
- Apply PCA to well aligned face images
- **W**: Eigenvectors in columns of matrix
- **S**: Eigenvalues along diagonal of matrix
- Note the 'global' properties



« Eigen decomposition»

- Si la matrice de données a été moyenne centrée
- La matrice de covariance est simplement

$$\Sigma = \mathbf{X}\mathbf{X}^T$$

- La matrice **W** contient les vecteurs singuliers, ou des vecteurs propres de la matrice de covariance

$$\mathbf{X}\mathbf{X}^T = \mathbf{W}\mathbf{S}\underbrace{\mathbf{D}^T\mathbf{D}}_{\mathbf{I}}\mathbf{S}^T\mathbf{W}^T = \mathbf{W}\mathbf{S}\mathbf{S}^T\mathbf{W}^T$$

- Les vecteurs propres avec la plus grande valeur propre ont la corrélation la plus forte avec des données

Eigendecomposition

- If the data matrix has been mean centered
- The covariance matrix is simply

$$\Sigma = \mathbf{X}\mathbf{X}^T$$

- The matrix **W** of singular vectors is equivalently the matrix **W** of eigenvectors of the observed covariance matrix

$$\mathbf{X}\mathbf{X}^T = \mathbf{W}\mathbf{S}\mathbf{S}^T\mathbf{W}^T$$

- Eigenvectors with the largest eigenvalue have the strongest correlation with data

Une représentation avec moins des dimensions

- Après que nous avons les matrices **W**, **S** et **D**
- La matrice des vecteurs de données X peut être transformée à Y avec moins de dimensions

$$\mathbf{Y} = \mathbf{W}^T\mathbf{X}$$

$$= \mathbf{S}\mathbf{D}^T$$

- Ex: PCA SIFT a une dimensionnalité originale de 3042
- Les expériences suggèrent que PCA SIFT puissent réaliser de bons résultats avec seulement 20 dimensions

The Lower Dimensional Representation

- Once we have matrices **W**, **S** and **D**
- The matrix of data vectors **X** can be transformed to lower dimensional **Y**

$$\mathbf{Y} = \mathbf{W}^T\mathbf{X}$$

$$= \mathbf{S}\mathbf{D}^T$$

- Example: PCA SIFT has input dimension 3042
- Experiments suggest that PCA SIFT can achieve good results with 20 dimensions

Une exemple dans l'analyse des documents (article de recherche célèbre) : « Latent Semantic Indexing » (LSI)

Technical Memo Example

Titles

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey of user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

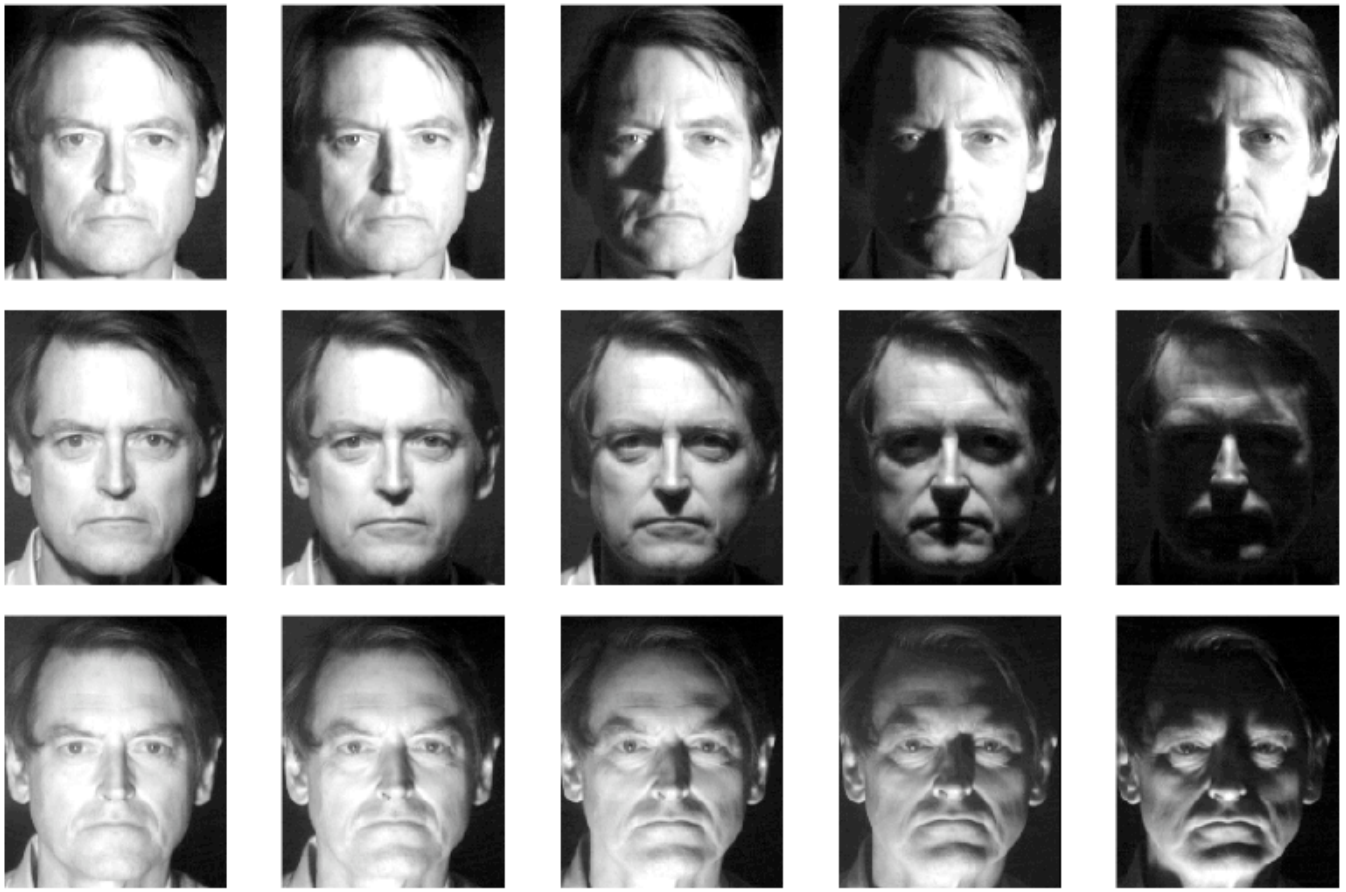
m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors: A survey*

“Eigen Documents”

.22	-.11	Human	**.29**	-.41	-.11	-.34
.20	-.07	Interface	.14	**.55**	**.28**	**.50**
.24	.04	Computer	-.16	-.59	-.11	-.25
.40	.06	User	-.34	.10	**.33**	**.38**
.64	-.17	System	**.36**	**.33**	-.16	-.21
.27	.11	Response	-.43	.07	.08	-.17
.27	.11	Time	-.43	.07	.08	-.17
.30	-.14	EPS	**.33**	**.19**	.11	.27
.21	.27	Survey	-.18	-.03	-.54	.08
.01	**.49**	Trees	.23	.03	**.59**	-.39
.04	**.62**	Graph	.22	.00	-.07	.11
.03	**.45**	Minors	.14	-.01	-.30	.28

- Eigenvector, Term, ‘Activity’ of Vector

Avec un base de donnée en consistant de plusieurs identités en forme suivant

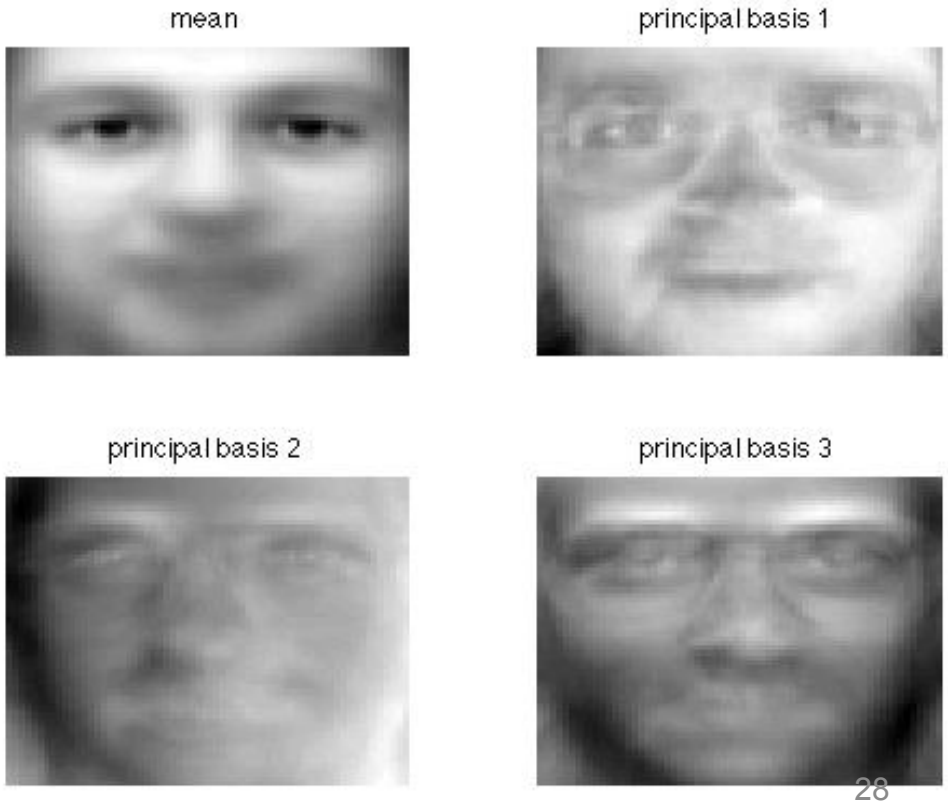


… il est facile à imaginer une interprétation pour les « eigenvisages »

« Eigenfaces » vs. « FisherFaces »

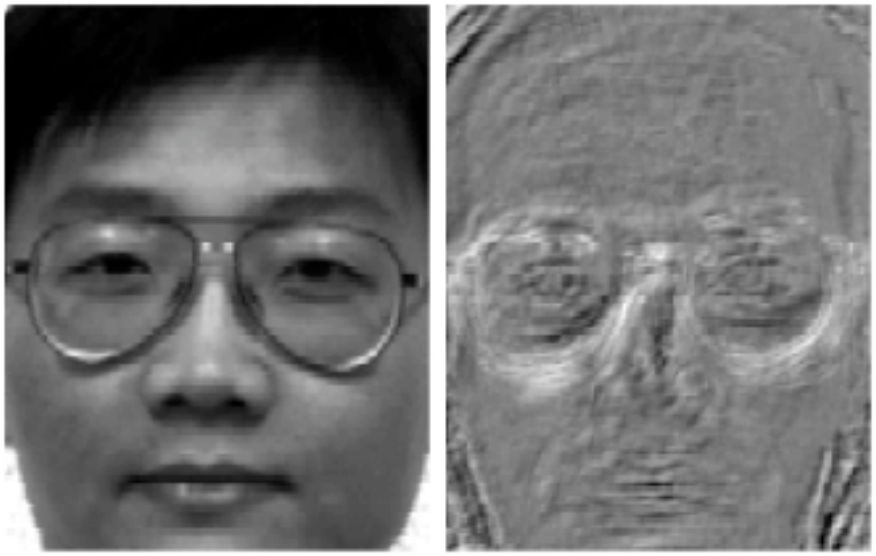
- Nous cherchons un détecteur pour les lunettes
- Toutefois avec ACP nous allons obtenir une réduction de la dimensionnalité avec un but différente

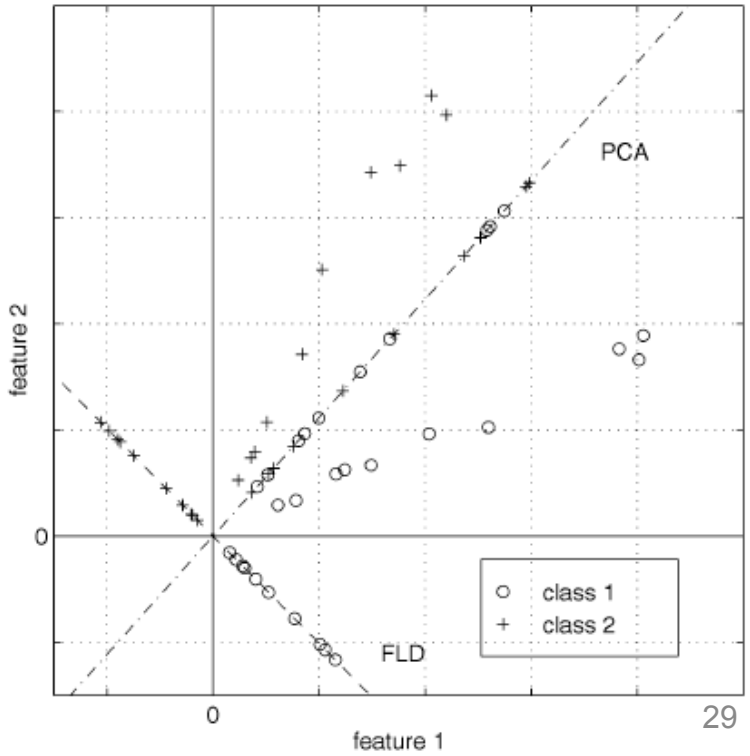
- ACP -> les dimensions de covariance maximale, ou
- Un modèle compact pour les caractéristiques, capable à capturer les variations dominantes



« Eigenfaces » vs. « FisherFaces »

- Imaginons que nous cherchons un détecteur pour les lunettes
- Voici un « Fisher Faces » et une comparaison entre PCA et FLD en 2D





De Belhumeur et al., PAMI 1997

L'analyse discriminante linéaire de Fisher

- Alternatif à l'ACP qui utilise les étiquettes

Nous maximisons $$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

avec $$\mathbf{S}_W = \sum_{k=1}^{K} \sum_{\bar{x}_j \in C_k} (x_j - \mu_k)(x_j - \mu_k)^T$$

et $$\mathbf{S}_B = \sum_{k=1}^{K} (\mu_k - \mu)(\mu_k - \mu)^T \qquad \mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

et quand k=2 $$\mathbf{S}_B = \sum_{k=1}^{K} (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

“Fisher’s LDA”

- On peut obtenir un matrice **W** de **w**s comme

$$\mathbf{W}_{opt} = \arg\max_{\mathbf{W}} \frac{\left|\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_w \mathbf{W}\right|} = \left[\mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_m\right]$$

où $\{\mathbf{w}_i \mid i=1,2,\ldots,m\}$ sont les les vecteurs propres généralisés de S_B et S_W qui correspondent avec le m valeurs propres λ_i le plus grandes, c.-à-d. ceux qui satisfont

$$\mathbf{S}_B \mathbf{w}_i = \lambda_i \mathbf{S}_w \mathbf{w}_i$$

LDA

LDA computes the projection α that maximizes the ratio:

$$\alpha_{opt} = argmax_{\alpha} \frac{|\alpha^T B \alpha|}{|\alpha^T W \alpha|}$$

by solving the generalized eigenvalue problem:

$$B\alpha = \lambda W \alpha$$

Data Mining

Practical Machine Learning Tools and Techniques

Slides for Chapter 8, Data transformations

of *Data Mining* by I. H. Witten, E. Frank,
M. A. Hall, and C. J. Pal

Projections

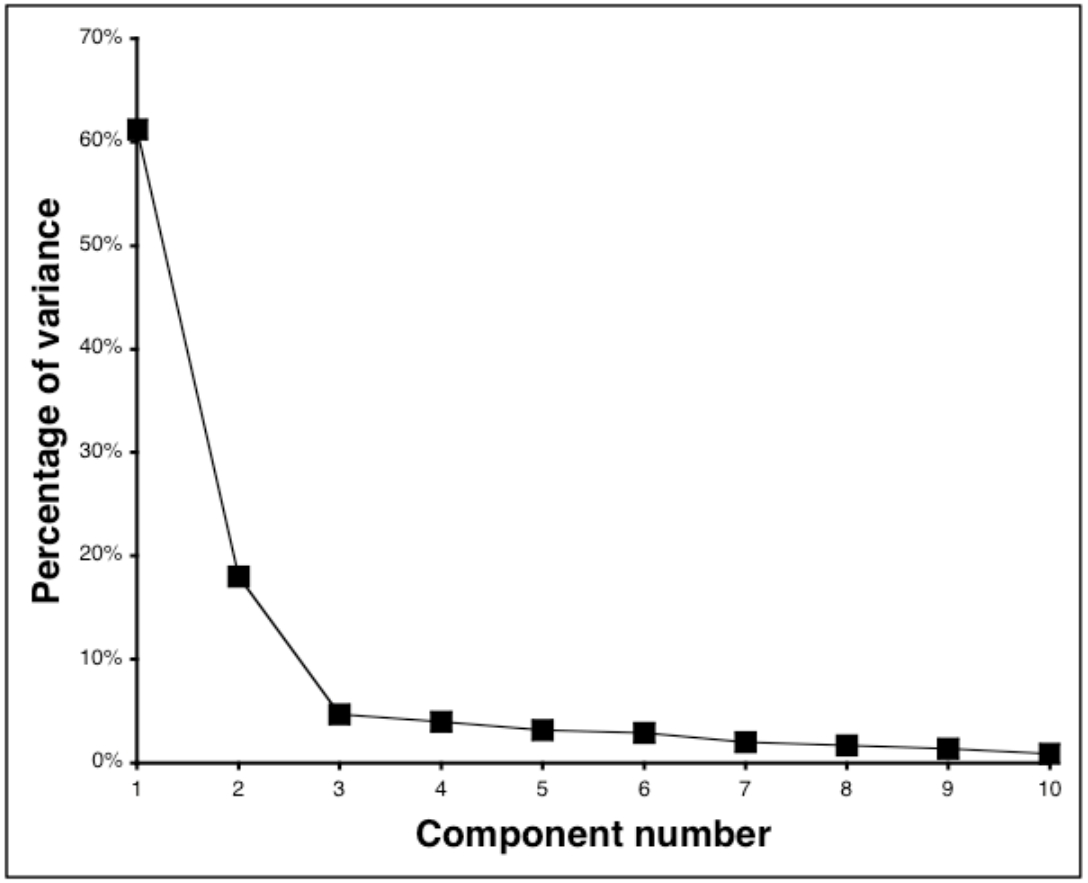
- Simple transformations can often make a large difference in performance
- Example transformations (not necessarily for performance improvement):
  - Difference of two date attributes
  - Ratio of two numeric (ratio-scale) attributes
  - Concatenating the values of nominal attributes
  - Encoding cluster membership
  - Adding noise to data
  - Removing data randomly or selectively
  - Obfuscating the data

Principal component analysis

- Unsupervised method for identifying the important directions in a dataset
- We can then rotate the data into the (reduced) coordinate system that is given by those directions
- PCA is a method for *dimensionality reduction*
- Algorithm:
  1. Find direction (axis) of greatest variance
  2. Find direction of greatest variance that is perpendicular to previous direction and repeat
- Implementation: find eigenvectors of the covariance matrix of the data
  - Eigenvectors (sorted by eigenvalues) are the directions

Example: 10-dimensional data

Axis	Variance	Cumulative
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100.0%



- Data is normally standardized or mean-centered for PCA
- Can also apply this recursively in a tree learner

Random projections

- PCA is nice but expensive: cubic in number of attributes
- Alternative: use random directions instead of principal components
- Surprising: random projections preserve distance relationships quite well (on average)
  - Can use them to apply *k*D-trees to high-dimensional data
  - Can improve stability by using ensemble of models based on different projections
- Different methods for generating random projection matrices have been proposed

Partial least-squares regression

- PCA is often used as a pre-processing step before applying a learning algorithm
  - When linear regression is applied, the resulting model is known as *principal components regression*
  - Output can be re-expressed in terms of the original attributes because PCA yields a linear transformation
- PCA is unsupervised and ignores the target attribute
- The p*artial least-squares* transformation differs from PCA in that it takes the class attribute into account
  - Finds directions that have high variance and are strongly correlated with the class
- Applying PLS as a pre-processing step for linear regression yields *partial least-squares regression*

An algorithm for PLS

1. Start with standardized input attributes
2. Attribute coefficients of the first PLS direction:
   - Compute the dot product between each attribute vector and the class vector in turn, this yields the coefficients
3. Coefficients for next PLS direction:
   - Replace attribute value by difference (residual) between the attribute's value and the prediction of that attribute from a simple regression based on the previous PLS direction
   - Compute the dot product between each attribute's residual vector and the class vector in turn, this yields the coefficients
4. Repeat from 3

Independent component analysis (ICA)

- PCA finds a coordinate system for a feature space that captures the covariance of the data
- In contrast, *ICA* seeks a projection that decomposes the data into sources that are statistically independent
- Consider the "cocktail party problem," where people hear music and the voices of other people: the goal is to un-mix these signals
- ICA finds a linear projection of the mixed signal that gives the most statistically independent set of transformed variables

Correlation vs. statistical independence

- PCA is sometimes thought of as a method that seeks to transform correlated variables into linearly uncorrelated ones
- Important: correlation and statistical independence are two different criteria
  - Uncorrelated variables have correlation coefficients equal to zero – entries in a covariance matrix
  - Two variables *A* and *B* are considered independent when their joint probability is equal to the product of their *marginal* probabilities:

$$P(A,B) = P(A)P(B)$$

ICA and Mutual Information

- Mutual information (MI) measures the amount of info one can obtain from one random variable given another one
- It can be used as an alternative criterion for finding a projection of data
  - We can aim to minimize the mutual information between the dimensions of the data in a linearly transformed space
- Assume a model s = Ax, where A is an orthogonal matrix, x is the input data and s is its decomposition into its sources
- Fact: minimizing the MI between the dimensions of s corresponds to finding a transformation matrix A so that
  a) the estimated probability distribution of the sources *p(s)* is as far from Gaussian as possible and
  b) the estimates s are constrained to be uncorrelated

ICA & FastICA

- A popular technique for performing independent component analysis is known as *fast ICA*
- Uses a quantity known as the *negentropy* J(s) = H(z) – H(s), where
  - z is a Gaussian random variable with the same covariance matrix as s and
  - H(.) is the "differential entropy," defined as

$$H(\mathbf{x}) = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}$$

- Negentropy measures the departure of s's distribution from the Gaussian distribution
- Fast ICA uses simple approximations to the negentropy allowing learning to be performed more quickly.

Linear discriminant analysis

- Linear discriminant analysis is another way of finding a linear transformation that reduces the number of dimensions
- Unlike PCA or ICA it uses labeled data: it is a supervised technique like PLS, but designed for classification
- Often used for dimensionality reduction prior to classification, but can be used as a classification technique itself
- When used for dimensionality reduction, it yields (k-1) dimensions for a k-class classification problem
- We will first discuss LDA-based classification (and, for completeness, QDA) before looking at data transformation

The LDA classifier

- Assumes data modeled by a multivariate Gaussian distribution for each class c, with mean μ_c and a common covariance Σ
- Assumption of *common* covariance matrix implies
  - the posterior distribution over the classes has a *linear* form and
  - there is a *linear* discriminant function for each class
- The linear discriminant classifier is computed as

$$y_c = \mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c - 1/2\,\boldsymbol{\mu}_c^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c + \log(n_c/n)$$

- where n_c is the number of examples of class *c* and *n* is the total number of examples
- Common variance matrix is obtained by pooling the covariance matrices from the different classes using a weighted average
- The data is classified by choosing the largest y_c

Quadratic discriminant analysis (QDA)

- The *QDA* classifier is obtained by giving each class its *own* covariance matrix Σ_c
- In QDA, the decision boundaries defined by the posterior over classes are described by quadratic equations
- The quadratic discriminant function for each class *c* is:

$$f_c(\mathbf{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_c| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)\boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)^{\mathrm{T}} + \log(n_c/n),$$

- These functions, as the ones for LDA, are produced by taking the log of the corresponding Gaussian model for each class
  - Constant terms can be ignored because such functions will be compared with one another

Fisher's linear discriminant analysis

- Let us now consider Fisher's LDA projection for dimensionality reduction, considering the two-class case first
- We seek a projection vector a that can be used to compute scalar projections y = ax for input vectors x
- This vector is obtained by computing the means of each class, μ_1 and μ_2, and then computing two special matrices
- The *between-class scatter matrix* is calculated as

$$\mathbf{S}_B = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathrm{T}}$$

(note the use of the outer product of two vectors here, which gives a matrix)

- The *within-class scatter matrix* is

$$\mathbf{S}_W = \sum_{i:c_i=1} (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^{\mathrm{T}} + \sum_{i:c_i=2} (\mathbf{x}_i - \boldsymbol{\mu}_2)(\mathbf{x}_i - \boldsymbol{\mu}_2)^{\mathrm{T}}$$

Fisher's LDA: the solution vector

- The solution vector a for FLDA is found by maximizing the "Rayleigh quotient"

$$J(\mathbf{a}) = \frac{\mathbf{a}^{\mathrm{T}}\mathbf{S}_B\mathbf{a}}{\mathbf{a}^{\mathrm{T}}\mathbf{S}_W\mathbf{a}}$$

- This leads to the solution

$$\mathbf{a} = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$$

Multi-class FLDA

- FLDA can be generalized to multi-class problems
- Aim: projection A so that $y = A^T x$ yields points that are close when they are in the same class relative to the overall spread
- To do this, first compute both the means for each class and the global mean, and then compute the scatter matrices

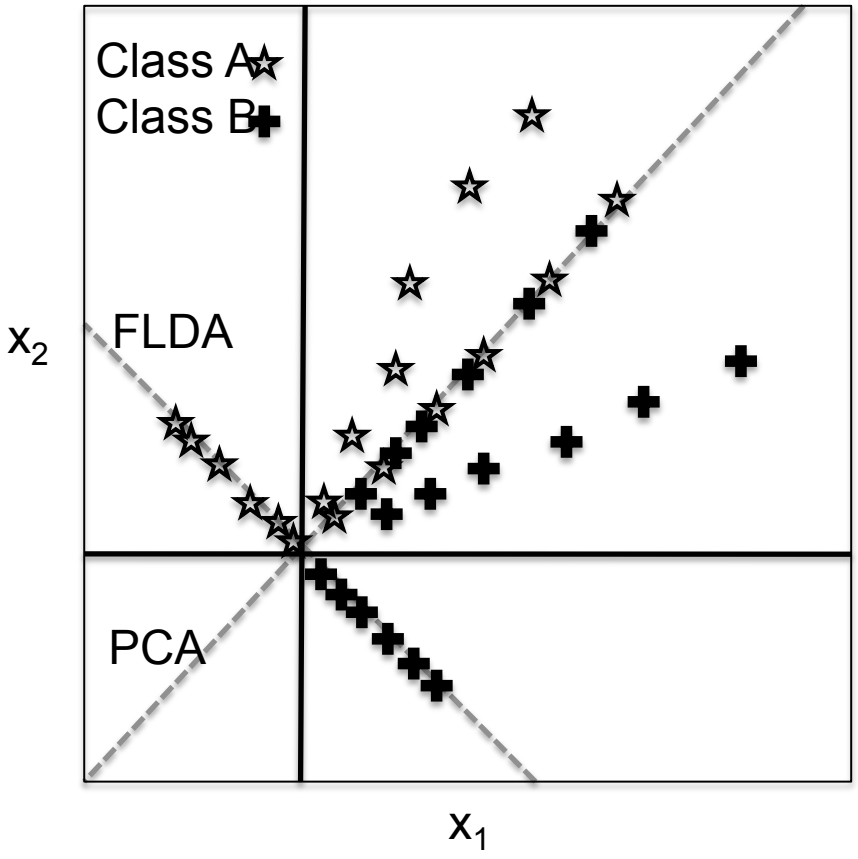
$$\mathbf{S}_B = \sum_{c=1}^{C} n_c (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^{\mathrm{T}} \qquad \mathbf{S}_W = \sum_{j=1}^{C} \sum_{i:c_i=j} (\mathbf{x}_i - \boldsymbol{\mu}_{c=j})(\mathbf{x}_i - \boldsymbol{\mu}_{c=j})^{\mathrm{T}}$$

- Then, find the projection matrix A that maximizes

$$J(\mathbf{A}) = \frac{\left| A^{\mathrm{T}} \mathbf{S}_B A \right|}{\left| A^{\mathrm{T}} \mathbf{S}_W A \right|}$$

- Determinants are analogs of variances computed in multiple dimensions, along the principal directions of the scatter matrices, and multiplied together
- Solutions for finding A are based on solving a "generalized eigenvalue problem" for each column of the matrix A.

Fisher's LDA vs PCA



Adapted from Belhumeur, Hespanha & Kriegman, 1997

Data Mining

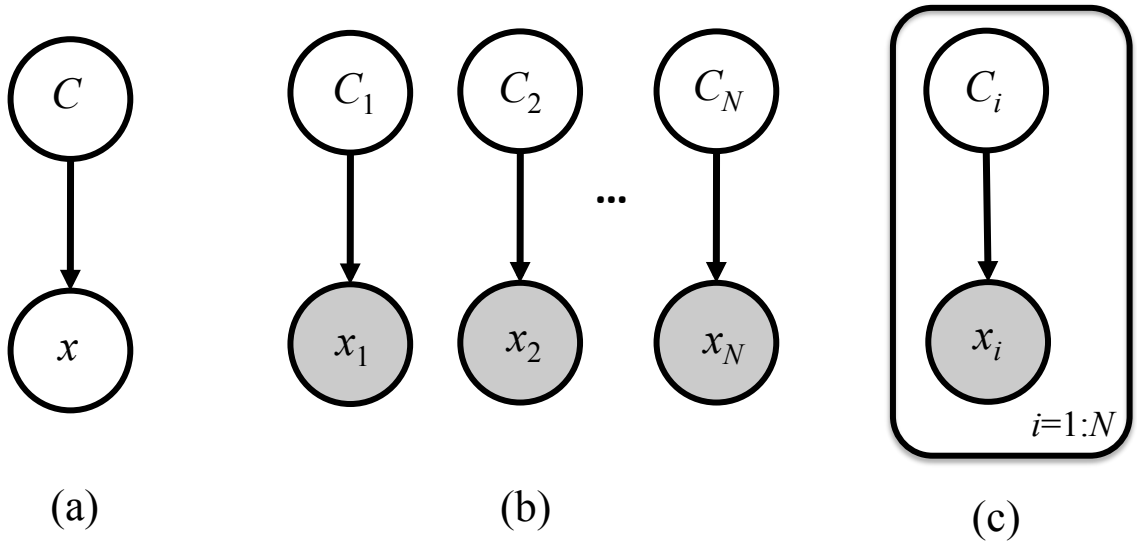
Practical Machine Learning Tools and Techniques

Slides for Chapter 9, Probabilistic methods

of *Data Mining* by I. H. Witten, E. Frank,
M. A. Hall, and C. J. Pal

Plate notation

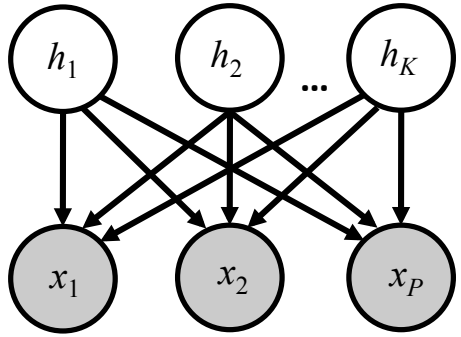
- A "plate" is simply a box around a Bayesian network that denotes a certain number of replications of it, one for each data instance
- The plate below indicates i=1…N networks, each with an observed value for x_i and hidden variable C_i
- Plate notation captures a model for the joint probability of the entire data with a simple picture.



Probabilistic PCA

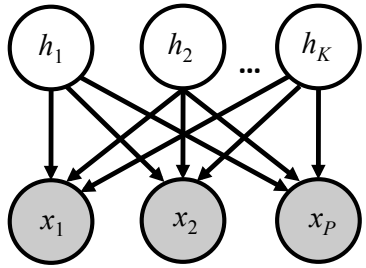
- Probabilistic PCA, (PPCA) can be written as a Bayes net
- Let **x** be a *d*-dimensional random vector of observed data, and **h** be a *k*-dimensional vector of hidden, $k < d$ typically
- The joint probability model has this linear Gaussian form

$$p(\mathbf{x},\mathbf{h}) = p(\mathbf{x}\,|\,\mathbf{h})p(\mathbf{h})$$
$$= N(\mathbf{x};\mathbf{W}\mathbf{h}+\boldsymbol{\mu},\mathbf{D})N(\mathbf{h};\mathbf{0},\mathbf{I}),$$



- where $p(\mathbf{h})$ is a Gaussian distributed random variable having a zero mean and identity covariance, while $p(\mathbf{x}|\mathbf{h})$ is Gaussian with mean $\mathbf{W}\mathbf{h}+\boldsymbol{\mu}$, and a diagonal covariance matrix $\mathbf{D}=\sigma^2\mathbf{I}$
- The mean $\boldsymbol{\mu}$ is included as a parameter, but it would be zero if we first mean-centered the data, mixtures of PPCA use these means to model more complex distributions

PPCA and Factor Analysis



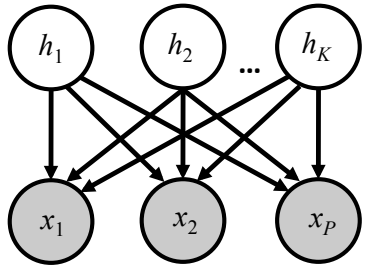
- Restricting the covariance matrix **D** to be diagonal produces a model known as *factor analysis*
- Because of the nice properties of Gaussian distributions, *the marginal distribution* of **x** in these models are also Gaussian

$$p(\mathbf{x}) = N(\mathbf{x};\boldsymbol{\mu},\mathbf{M} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I})$$

- The *posterior distribution* for **h** can be obtained from Bayes' rule, and some Gaussian identities

$$p(\mathbf{h} \mid \mathbf{x}) = N(\mathbf{h};\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}),\sigma^2\mathbf{M}^{-1})$$

PPCA, learning and the marginal likelihood



- The *log marginal likelihood* of the parameters given all the observed data can be maximized using this objective function

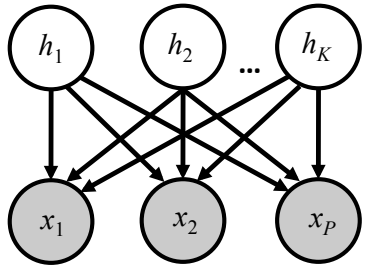
$$L(\tilde{\mathbf{X}};\theta) = \log\left[\prod_{i=1}^{N} P(\tilde{\mathbf{x}}_i;\theta)\right] = \sum_{i=1}^{N} \log\left[N(\tilde{\mathbf{x}}_i;\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\mathbf{T}} + \sigma^2\mathbf{I})\right]$$

where the model parameters are $\theta = \{\mathbf{W}, \boldsymbol{\mu}, \sigma^2\}$

- Compare this to the joint probability

$$L(\tilde{\mathbf{X}}, \mathbf{H};\theta) = \log\left[\prod_{i=1}^{N} p(\tilde{\mathbf{x}}_i, \mathbf{h}_i;\theta)\right] = \sum_{i=1}^{N} \log\left[p(\tilde{\mathbf{x}}_i \mid \mathbf{h}_i; \mathbf{W}) p(\mathbf{h}_i;\sigma^2)\right].$$

Gradient ascent for PPCA



- The expected log-likelihood (ELL) of the data is

$$\mathrm{E}[L(\tilde{\mathbf{X}},\mathbf{H};\theta)]_{p(\mathbf{H}|\tilde{\mathbf{X}})} = \sum_{i=1}^{N} \mathrm{E}\Big[\log\big[p(\tilde{\mathbf{x}}_i,\mathbf{h}_i;\theta)\big]\Big]_{p(\mathbf{h}_i|\tilde{\mathbf{x}}_i)}$$

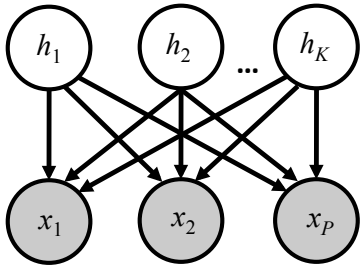
- The gradient of the ELL under $p(\mathbf{h} \mid \tilde{\mathbf{x}})$ is

$$\frac{\partial}{\partial \mathbf{W}} \mathrm{E}[L(\tilde{\mathbf{x}},\mathbf{h})] = \mathrm{E}[\mathbf{W}\mathbf{h}\mathbf{h}^{\mathrm{T}}] - \mathrm{E}[\tilde{\mathbf{x}}\mathbf{h}^{\mathrm{T}}]$$

$$= \mathbf{W}\,\mathrm{E}[\mathbf{h}\mathbf{h}^{\mathrm{T}}] - \tilde{\mathbf{x}}\,\mathrm{E}[\mathbf{h}]^{\mathrm{T}}$$

- This has a natural interpretation as a difference between two expectations, the first is akin to the models prediction or reconstruction of the input forming a matrix with the models explanation of the input

EM for PPCA



- The E and M steps of the PPCA EM algorithm can be written as

E-Step:
$$\mathrm{E}[\mathbf{h}_i] = \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}\tilde{\mathbf{x}}_i, \quad \mathrm{E}[\mathbf{h}_i\mathbf{h}_i^{\mathrm{T}}] = \sigma^2\mathbf{M}^{-1} + \mathrm{E}[\mathbf{h}_i]\,\mathrm{E}[\mathbf{h}_i^{\mathrm{T}}],$$

M-Step:
$$\mathbf{W}^{New} = \left[\sum_{i=1}^{N} \tilde{\mathbf{x}}_i\, \mathrm{E}[\mathbf{h}_i]^{\mathbf{T}}\right]\left[\sum_{i=1}^{N} \mathrm{E}[\mathbf{h}_i\mathbf{h}_i{}^{\mathrm{T}}]\right]^{-1},$$

where all expectations are taken with respect to each example's posterior distribution

- In the zero input noise limit these equations can be written even more compactly

Latent semantic analysis (LSA) and the singular value decomposition (SVD)

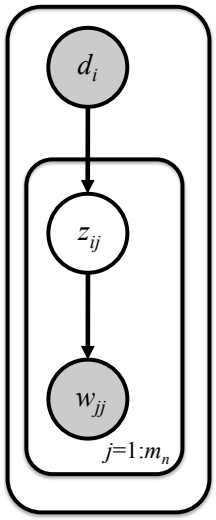
- LSA factorizes a data matrix (ex. of word counts) using SVD

$$\underset{t \times d}{\left[\tilde{\mathbf{X}}\right]} = \underset{t \times k}{\left[\mathbf{U}_k\right]} \underset{k \times k}{\begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_k \end{bmatrix}} \underset{k \times d}{\left[\mathbf{V}_k^{\mathrm{T}}\right]}$$

where **U** and **V** have orthogonal columns and **S** is a diagonal matrix containing the singular values, usually sorted in decreasing order (t=#terms, d=#docs, k=#topics)

- For every $k < d$, $\tilde{\mathbf{X}} \approx \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^{\mathrm{T}}$ and if all but the k largest singular values are discarded the data matrix can be reconstructed in an optimal in a least squares sense

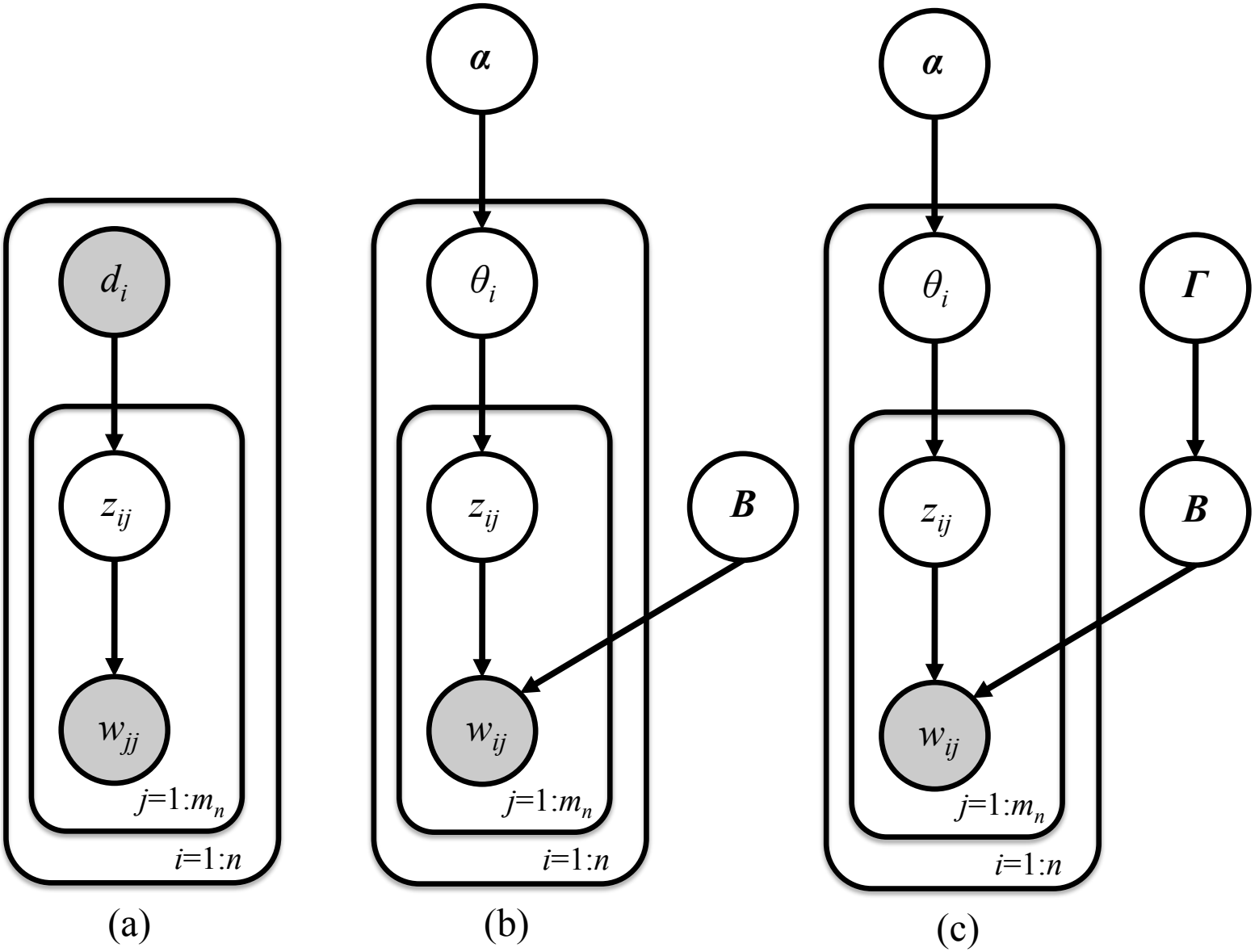
Probabilistic Latent Semantic Analysis (pLSA)



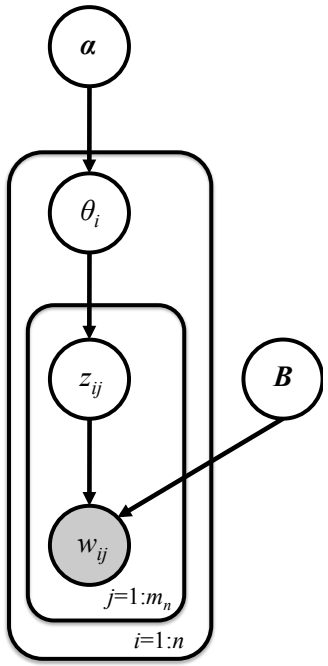
- In the pLSA framework one considers the *index* of each document as encoded using observations of discrete random variables d_i for i=1,…,n documents
- Each variable d_i has n states, and over the document corpus there is one observation of the variable for each state.
- Topics represented with discrete variables z_{ij}, while words are represented with random variables w_{ij}, where m_i words are associated with each document and each word is associated with a topic.

$$P(W,D) = \prod_{i=1}^{n} P(d_i) \prod_{j=1}^{m_n} \sum_{z_{ij}} P(z_{ij} \mid d_i) P(w_{ij} \mid z_{ij}).$$

pLSA, LDA and smoothed LDA



Latent Dirichlet Allocation (LDA)



- Reformulates pLSA replacing document index variables d_i with the random parameter θ
- The distribution of θ is influenced by a Dirichlet prior with hyperparameter α
- The relationship between discrete topic variables z_{ij} and the words w_{ij} is also given an explicit dependence on the hyperparameter, matrix **B** .
- The probability model for all observed words W is

$$P(\mathrm{W}|\boldsymbol{\alpha},\mathbf{B}) = \prod_{i=1}^{n} \int P(\boldsymbol{\theta}_i \mid \boldsymbol{\alpha}) \left[\prod_{j=1}^{m_n} \sum_{z_{ij}} P(z_{ij} \mid \boldsymbol{\theta}_i) P(w_{ij} \mid z_{ij}, \mathbf{B}) \right] d\boldsymbol{\theta}_i$$

$$= \prod_{i=1}^{n} \int P(\boldsymbol{\theta}_i \mid \boldsymbol{\alpha}) \left[\prod_{j=1}^{m_n} P(w_{ij} \mid \boldsymbol{\theta}_i, \mathbf{B}) \right] d\boldsymbol{\theta}_i,$$

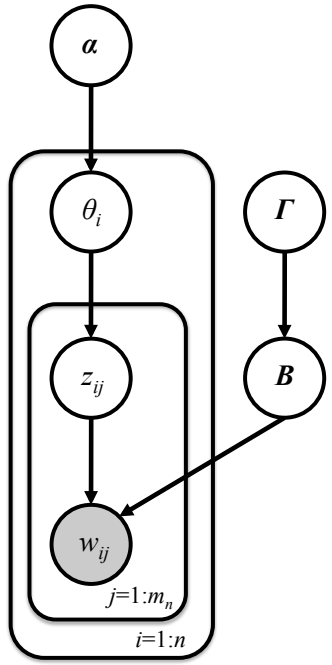
Inference for LDA

- The marginal log-likelihood of the model can be optimized using an empirical Bayesian method by adjusting the hyperparameter α and parameter B
- To perform the E-step of EM, the posterior distribution over the unobserved random quantities is used

$$P(\boldsymbol{\theta},\mathbf{z}|\mathbf{w},\boldsymbol{\alpha},\mathbf{B}) = \frac{P(\boldsymbol{\theta},\mathbf{z},\mathbf{w}|\boldsymbol{\alpha},\mathbf{B})}{P(\mathbf{w}|\boldsymbol{\alpha},\mathbf{B})}$$

- This posterior is intractable and is typically computed using a variational approximation or by sampling

Smoothed LDA



- Reduces the effects of overfitting
- Adds another Dirichlet prior with hyperparameters given by **Γ** on the topic parameters **B**
- *Collapsed Gibbs sampling* can be performed by integrating out the θs and **B** analytically, which deals with these distributions *exactly*
- The Gibbs sampler proceeds by simply iteratively updating each z_{ij} conditioned on **Γ** and α to compute the required approximate posterior

Topics from PNAS

- Example: Griffiths and Steyvers (2004) applied smoothed LDA to 28,154 abstracts of papers published in the *Proceedings of the National Academy of Science (PNAS)* from 1991 to 2001

Topic 2	**Topic 39**	**Topic 102**	**Topic 201**	**Topic 210**
Species	Theory	Tumor	Resistance	Synaptic
Global	Time	Cancer	Resistant	Neurons
Climate	Space	Tumors	Drug	Postsynaptic
Co_2	Given	Human	Drugs	Hippocampal
Water	Problem	Cells	Sensitive	Synapses
Geophysics, geology, ecology	Physics, math, applied math	Medical sciences	Pharmacology	Neurobiology

- User tags shown at the bottom were not used to create the topics, but correlate very well with the inferred topics

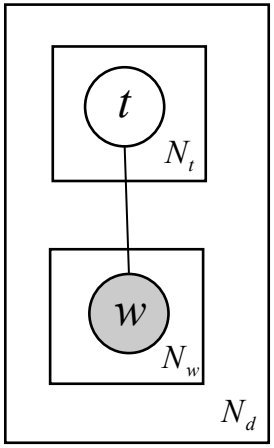
Boltzmann Machines and ‘Harmonium’ Based Methods

Chris Pal

Contrast - MCA, PCA, RAP

- Multinomial Component Analysis (MCA)
- Principal Component Analysis (PCA)
- Rate Adapting Poisson (RAP) Model

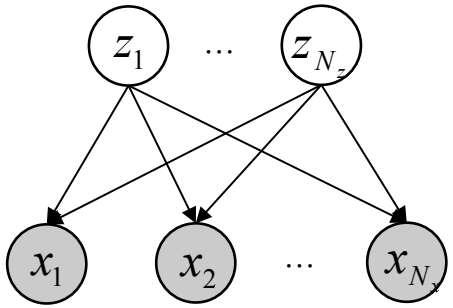
MCA



N_w draws from a discrete distribution, (words in doc)

PCA

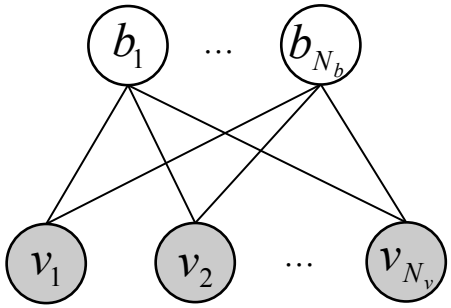
N_z unobserved, Gaussian variables



N_x observed, Gaussian variables, fixed dimension

RAP

N_b binary topics

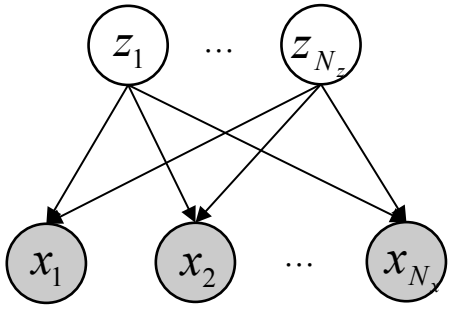


N_v Poisson counts for each word in vocabulary

ACP et un machine de Boltzmann comme des modèles probabilistes

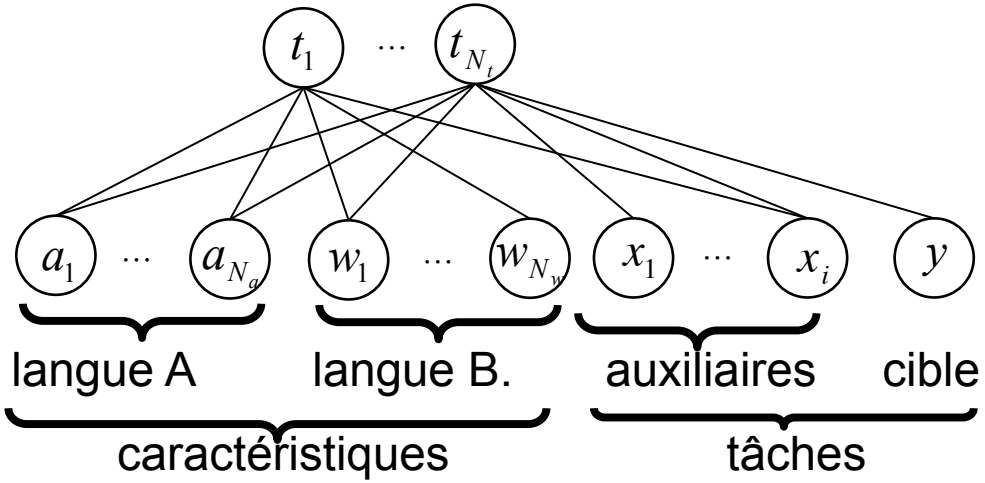
ACP / PCA

N_z variables inobservées,
P(z) ~ Gaussienne



N_x >> Nz variables observées
P(x|z) ~ linéaire gaussien

N_z variables inobservées



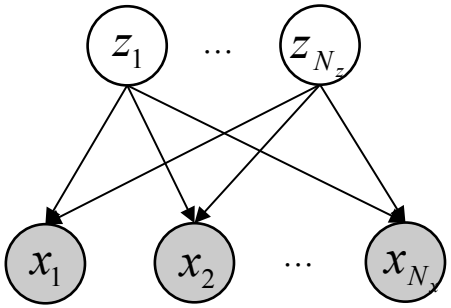
- Analyse en composantes principales (ACP)
- Généralisations avec les modèles probabilistes, les machines de Boltzmann et apprentissage discriminatif vs générative

Boltzmann Machines & Some New Continuous Topic Models

- Undirected (Random Field) Joint Model
- Conditionally log-Normal Topics
- Conditionally Multinomial Words

PCA

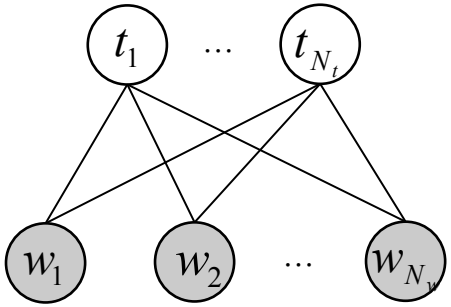
N_z unobserved, Gaussian variables



N_x observed, Gaussian variables, fixed dimension

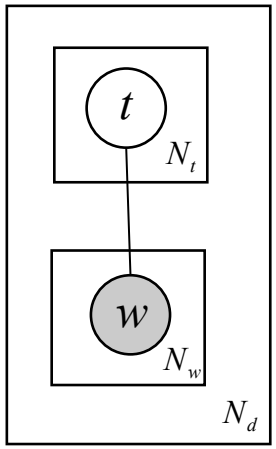
Boltzmann Machine / Champ aleatoire

N_t topics

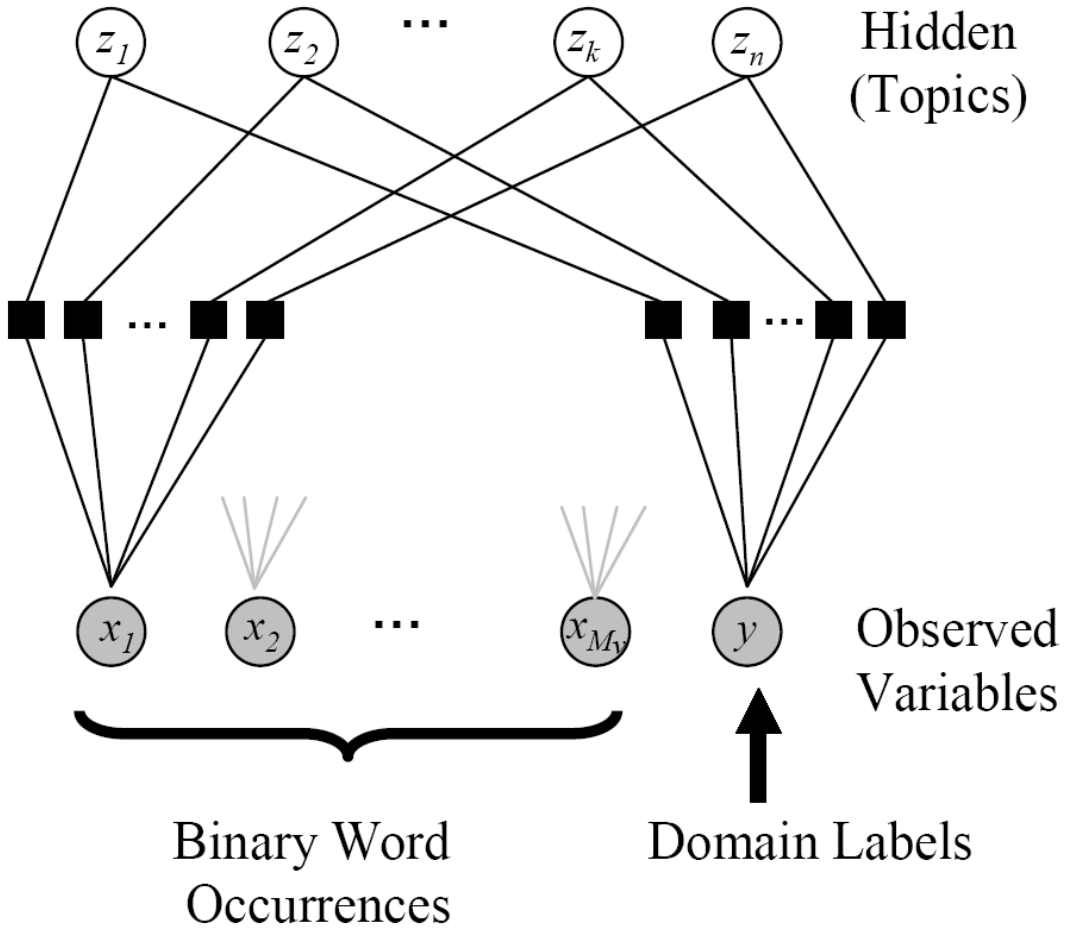


N_w words

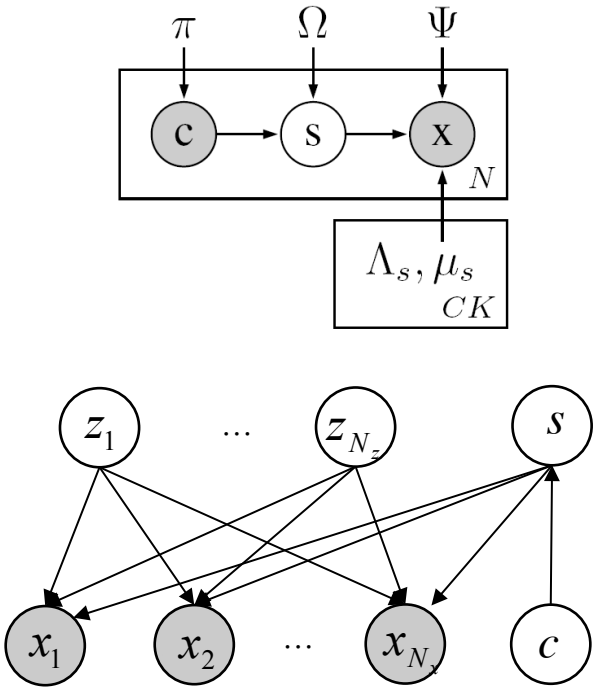
Plate Notation

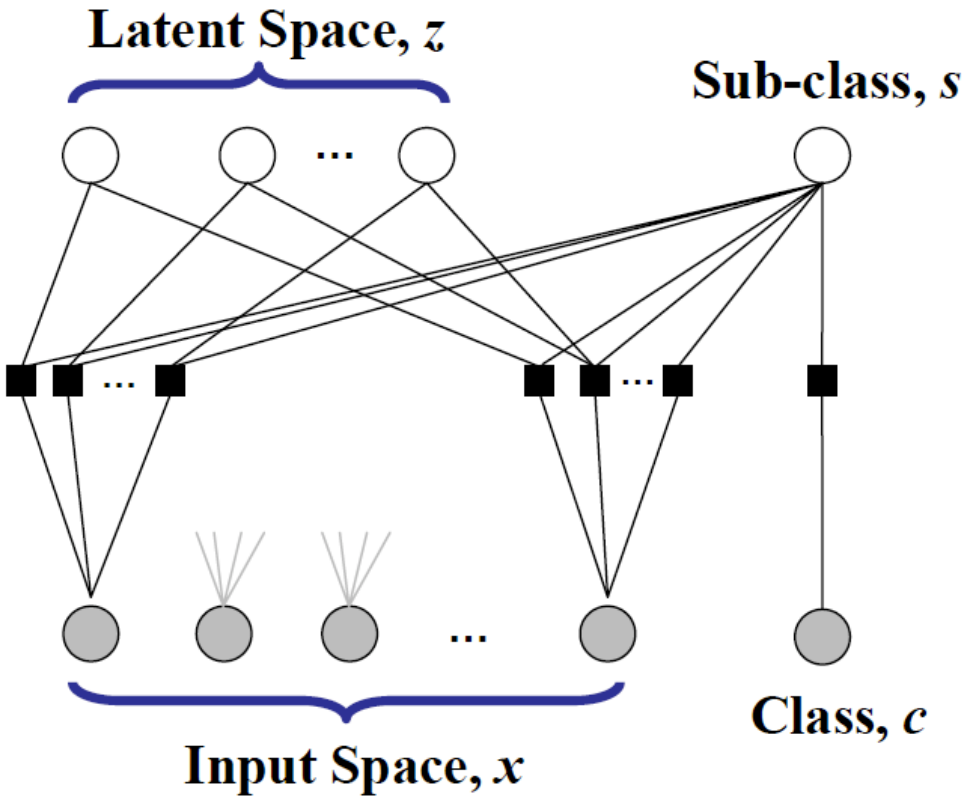


Restricted Boltzmann Machines – Viewed Here as a Factor Graph



Compare with Mixtures of Factor Analyzers





The Main Equations

- The conditionals for Gibbs sampling

$$\begin{aligned} P(\mathbf{z}_n|\tilde{\mathbf{x}}) &= \mathcal{N}(\mathbf{z}_n;\hat{\boldsymbol{\mu}},\mathbf{I}), \quad \hat{\boldsymbol{\mu}} = \boldsymbol{\mu} + \mathbf{W}^T\tilde{\mathbf{x}} \\ P(\mathbf{x}_b|\tilde{\mathbf{z}}) &= \mathcal{B}(\mathbf{x}_b;\hat{\boldsymbol{\theta}}_b), \quad \hat{\boldsymbol{\theta}}_b = \boldsymbol{\theta}_b + \mathbf{W}_b\tilde{\mathbf{z}} \\ P(\mathbf{x}_d|\tilde{\mathbf{z}}) &= \mathcal{D}(\mathbf{x}_d;\hat{\boldsymbol{\theta}}_d), \quad \hat{\boldsymbol{\theta}}_d = \boldsymbol{\theta}_d + \mathbf{W}_d\tilde{\mathbf{z}} \end{aligned}$$

- Optimize the marginal or marg. conditionals

$$P(\mathbf{x}|\boldsymbol{\theta},\boldsymbol{\Lambda}) = \exp\{\boldsymbol{\theta}^T\mathbf{x} + \mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x} - A(\boldsymbol{\theta},\boldsymbol{\Lambda})\} \qquad \boldsymbol{\Lambda} = \tfrac{1}{2}\mathbf{W}\mathbf{W}^T$$

- Optimize the marginal or marg. conditionals

$$\frac{\partial\mathcal{L}(\theta;\mathbf{x})}{\partial\theta} = N\left[\left\langle\frac{\partial\mathbf{F}(\mathbf{x};\theta)}{\partial\theta}\right\rangle_{\tilde{P}(\mathbf{x})} - \left\langle\frac{\partial\mathbf{F}(\mathbf{x};\theta)}{\partial\theta}\right\rangle_{P(\mathbf{x};\theta)}\right]$$

Échantillonnage de Gibbs

Fonction GIBBS(X = {Z, E=e} , *bn*)
retourne un échantillonne de P(Z|E=e)

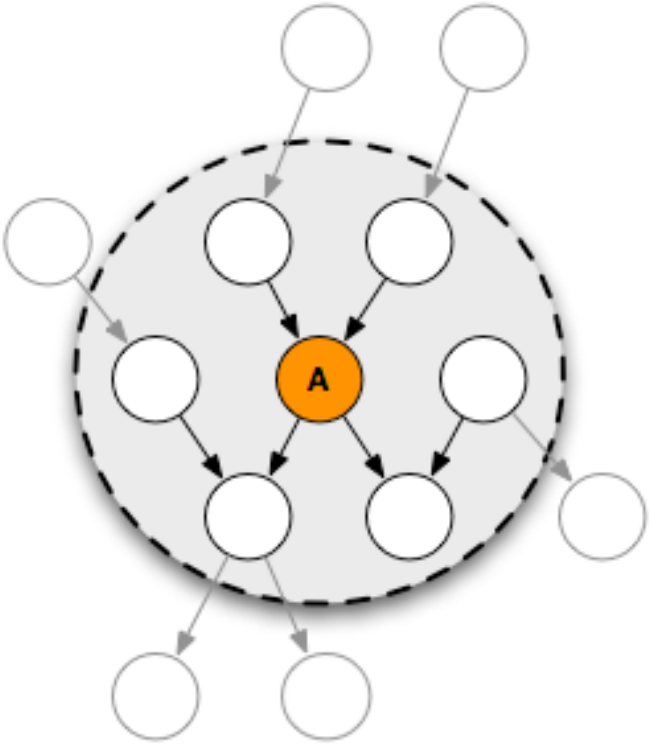
Variables locales:
Z, les variables non observées dans bn
x, l'état courant du réseau

Initialiser x = {z, e} avec des valeurs aléatoire pour Z=z

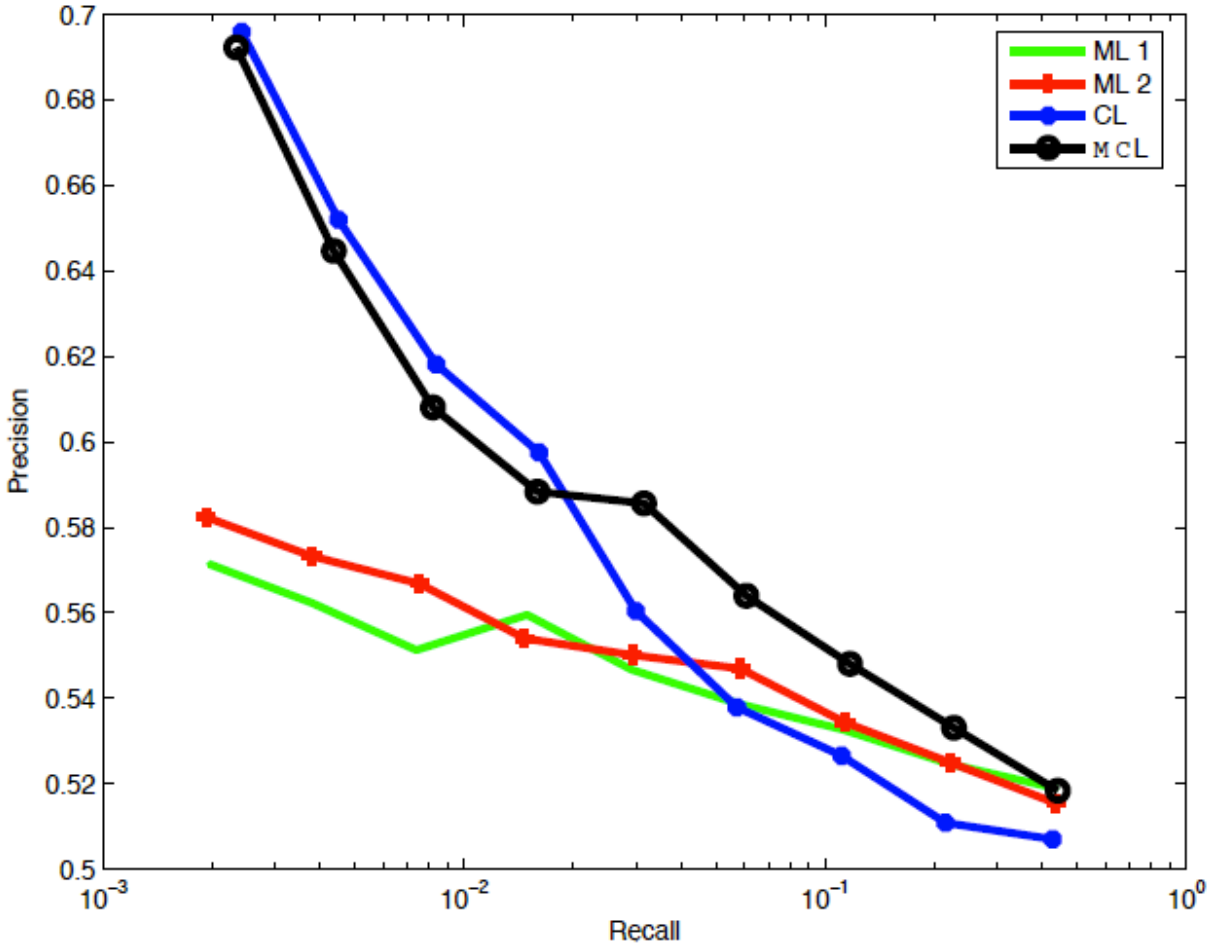
Pour j=1 à N faire
pour chaque Z_i dans Z faire
calculer la valeur de Zi dans x
par échantillonnage de $P(Z_i|mb(Z_i))$

Rappel: couverture de Markov

- Etant donné un nœud, la couverture de Markov consiste de ses parents, ses fils et les parents de ses fils.

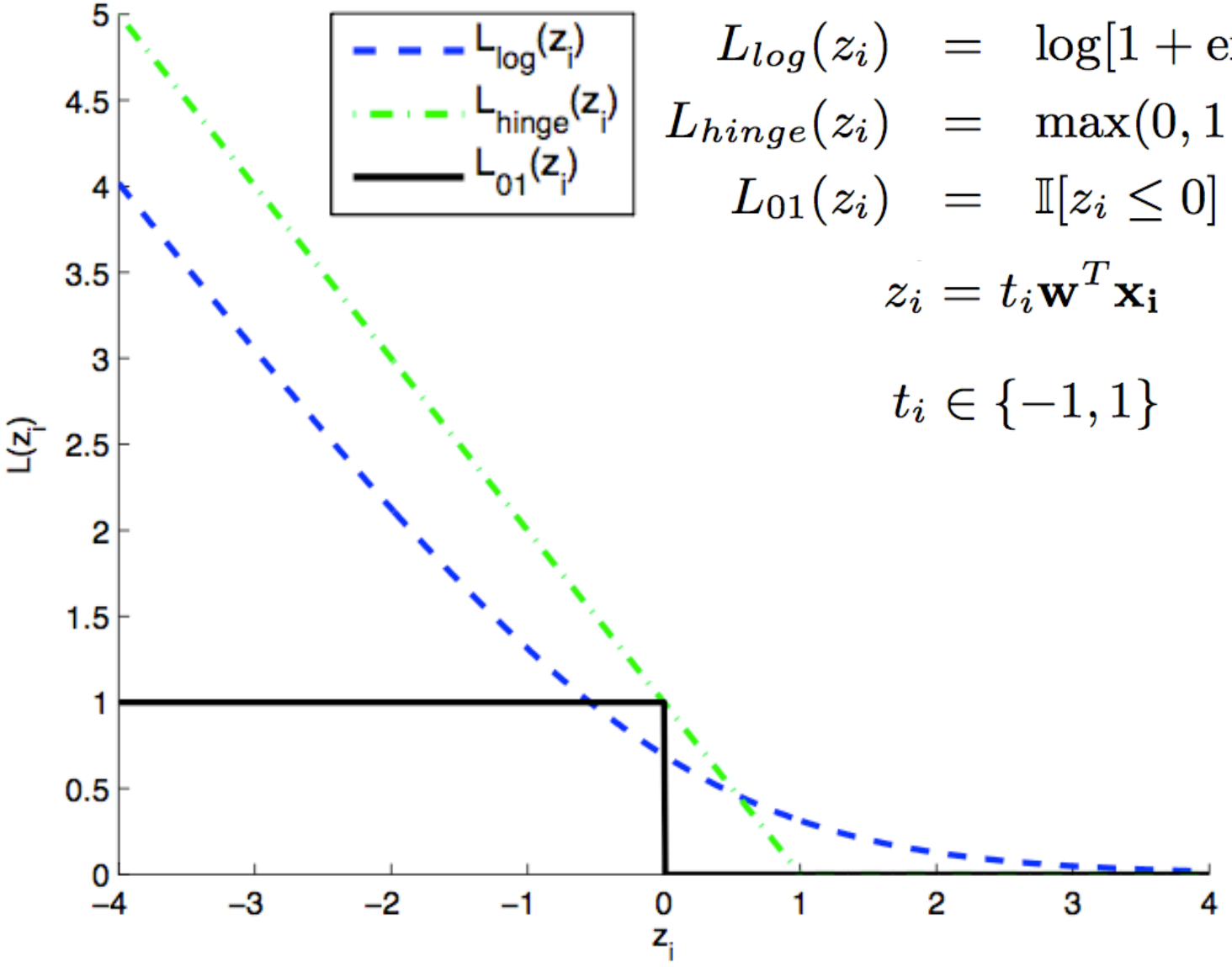


Precision vs Recall on 20 Newsgroups using RBM



Machines à vecteurs de support
« Support Vector Machines »

Comparez les « loss functions »



$$L_{log}(z_i) = \log[1 + \exp(-z_i)]$$

$$L_{hinge}(z_i) = \max(0, 1 - z_i)$$

$$L_{01}(z_i) = \mathbb{I}[z_i \leq 0]$$

$$z_i = t_i \mathbf{w}^T \mathbf{x_i}$$

$$t_i \in \{-1, 1\}$$

Minimisation de la « hinge loss »

$$\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\| + C\sum_{i=1}^{N} \max(0,1 - t_i(\mathbf{w}^T\mathbf{x}_i + w_0))$$

En ajoutant un « slack variable », ξ,

$$\min_{\mathbf{w},w_0,\xi} \frac{1}{2}\|\mathbf{w}\| + C\sum_{i=1}^{N} \xi_i \quad \text{(t.q/s.t.)}$$

$$\xi_i \geq 0, \quad \mathrm{t_i}(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \quad i = 1:N$$

Puis, on a un programme quadratique avec N+D+1 variables sujet à O(N) contraintes.

Machines à vecteurs de support

- Les SVMs sont actuellement l'approche la plus populaire pour l'apprentissage supervisé pour plusieurs problèmes à cause de leur performance supérieur
- Il y a trois propriétés qui les rendent séduisantes

Propriétés séduisantes

- Les SVMs construisent un **séparateur à marge maximale**, un frontière de décision avec la plus grande distance possible entre les points d'apprentissage.
- Cela leur permet de bien généraliser
- C'est un problème convexe – **avec un seule minima globale**

Propriétés séduisantes

- Les SVMs créent un hyperplan (linéaire séparateur), mais sont capables de projeter les données dans un espace de dimension plus grande en utilisant ce qu'on appelle **l'astuce du noyau**
- Il arrive souvent que des données qui ne sont pas linéairement séparables dans l'espace d'entrée d'origine le soient dans un espace de plus grande dimension
- Le séparateur linéaire de l'espace de plus grande dimension est en fait **non linéaire** *dans l'espace d'origine*

Propriétés séduisantes

- Les SVMs sont une méthode non paramétrique : elles mémorisent tous les exemples d'apprentissage et peuvent avoir besoin de tous les stocker
- Toutefois, souvent en pratique après l'optimisation de notre fonction d'objective, notre SVM va utiliser seulement un petit nombre d'exemples – les vecteurs de support – pour créer notre classificateur
- Donc, ils nous permet de représenter des fonctions complexes, mais ils sont résistantes au surapprentissage

Formulation des SVMs

- Nous utilisons les valeurs de +1 et -1 pour les étiquettes y_i
- Nous utilisons un vecteur de poids **w** pour les valeurs des paramètres
- Le séparateur correspondre avec un hyperplan, défini par { **x** : f(**x**) = $\mathbf{w}^T$ **x** + b = 0}
- La règle de classification : G(**x**) = signe($\mathbf{w}^T$**x** + b)
- Pour les données séparable $y_i f(\mathbf{x}_i) > 0 \quad \forall i$

Le séparateur à marge maximale

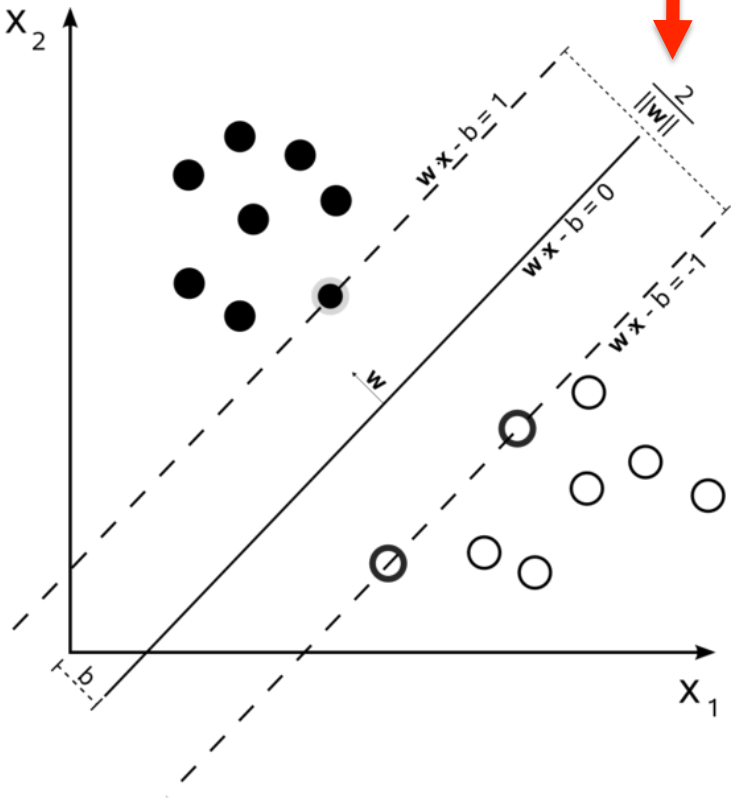
- Nous cherchons l'hyperplan ou **le séparateur à marge maximale –** ou la largeur de la zone délimitée dans la figure, c.-à-d. la distance de la droite séparatrice au point exemple le plus proche

Avec $C = \frac{1}{\|\mathbf{w}\|}$

Nous avons un problème d'optimisation de

$\max_{\mathbf{w},b,\|\mathbf{w}\|=1} C$ tels que

$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C, \quad i = 1,\ldots,N$



Formulation d'une Lagrangienne

- On peut réécrire le problème comme

$$\min_{\mathbf{w},b} \|\mathbf{w}\| \quad \text{tels que}$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1,\ldots,N$$

- Nous utilisions les multiplicateurs de Lagrange, $\alpha_n \geq 0$, un pour chaque contraint, qui nous donne une fonction Lagrangienne de

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}\alpha_i\left\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right\}$$

$$\left\{\frac{\partial L}{\partial \mathbf{w}} = 0, \frac{\partial L}{\partial a} = 0\right\} \rightarrow w = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i, \sum_{i=1}^{N}\alpha_i y_i = 0$$

La « représentation duale »

- L'élimination de w et b et avec ces conditions nous donne

$$\arg\max_{\alpha} \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_i - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k y_j y_k \mathbf{x}_j^T \mathbf{x}_k$$

$$\alpha_i, \xi_{\mathrm{i}} \geq 0 \quad \forall i, \ \sum_{\mathrm{j}} \alpha_j y_j = 0 \quad \forall j$$

- Ce qui donne les multiplicateurs de Lagrange optimaux α_j^* et nous avons $\mathbf{w} = \sum_{\mathrm{j}} \alpha_{\mathrm{j}} \mathbf{x}_j$
- C'est un problème d'optimisation de <u>programmation quadratique</u>

Les vecteurs de support

- Afin d'obtenir l'hyperplan solution, on remplace **w** par sa valeur optimale **w***

$$h(\mathbf{x}) = \text{signe}\left(\sum_{\text{j}} \alpha^*_j y_j (\mathbf{x}^T \mathbf{x}_j) - b\right)$$

- Et après l'optimisation nous avons une autre propriété fondamentale – les poids α_j associé à chaque point sont nuls sauf pour les vecteurs de support, c.-à-d. pour les points les plus proches du séparateur
- Les vecteurs de support sont généralement beaucoup moins nombreux que les exemples

La formulation standard

- Rappel : on peut écrire le problème comme

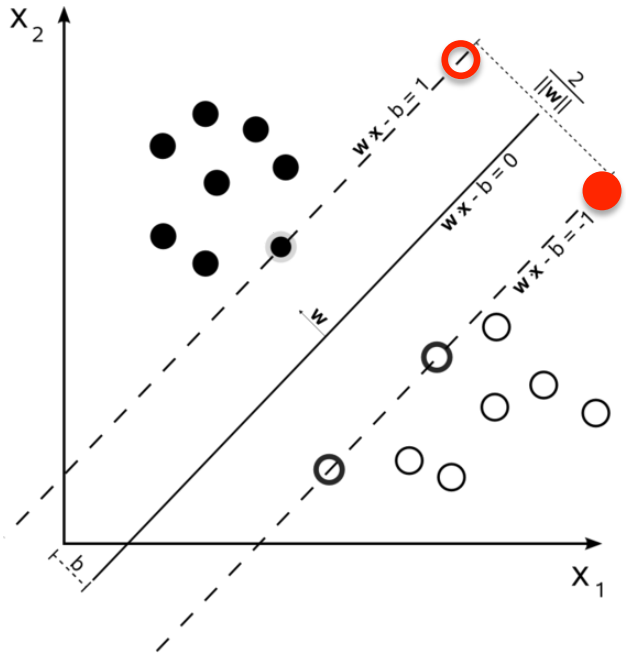
$$\min_{\mathbf{w},b} \|\mathbf{w}\| \quad \text{tels que}$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1,\ldots,N$$

- Que faire si nos données ne sont pas linéairement séparable?
- On peut modifier les contraintes comme

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C(1 - \xi_i),$$

Les marges
MVS vs. RL



$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|}, \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C(1-\xi_i), \quad \forall \mathrm{i}$$

Comparez avec la même idée dans le contexte de la RL

$$\max\left[\sum_{i=1}^{N}\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - \log Z(\mathbf{w},x_i)\right)\right]$$

La formulation du cas inséparable

- On peut formuler le problème comme

$$\min_{\mathbf{w},b,\xi} \|\mathbf{w}\| \qquad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C(1-\xi_i), \quad \forall \mathrm{i},$$
$$\xi_\mathrm{i} \geq 0, \sum \xi_\mathrm{i} \leq const.$$

- Il correspondre (comme le cas séparable) avec un problème d'optimisation quadratique et convexe
- Depuis $\xi_\mathrm{i} \geq 1$ implique que point i est en erreur, on pourrait interpréter $\sum_i \xi_\mathrm{i}$ comme une limite supérieure sur les erreurs
- Paramètre C donc control le nombre des erreurs
- Donc il est typique de sélectionner la valeur de C pendant une phase de validation croisée

La régression logistique avec l'astuce de noyau (RBF)

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\mathbf{y}^T \theta \mathbf{x}\right) \rightarrow$$

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\mathbf{y}^T \theta \mathbf{f}(\mathbf{x})\right), f_j(\mathbf{x}_i) = K(\mathbf{x}_i, \mathbf{x}_j)$$

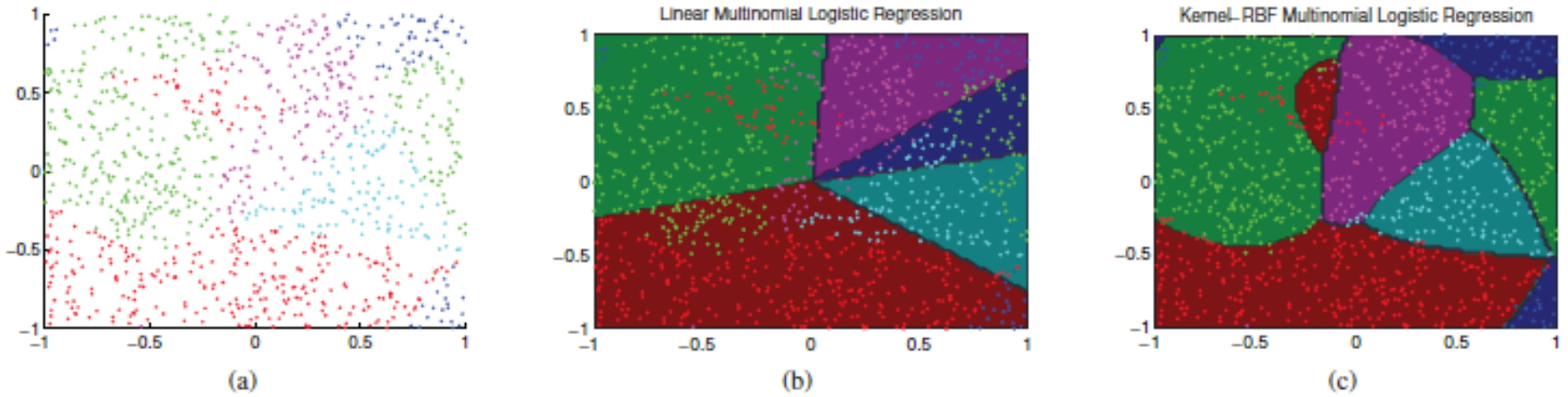


Figure 1.10: (a) Some 5 class data in 2d. (b) Multinomial logistic regression in the original feature space. (c) RBF basis functions with bandwidth of 1. We use all the data points as centers. Figure generated by `logregMultinomKernelDemo`.

L'astuce du noyau

$$h(\mathbf{x}) = \text{signe}\left(\sum_{\text{j}} \alpha^*_j y_j (\mathbf{x}^T \mathbf{x}_j) - b\right)$$

- Exemple: étant donné notre modèle linéaire, on peut remplacer $\mathbf{x}^T_i\mathbf{x}_j$ par $\mathbf{f}(\mathbf{x}_i)^T\mathbf{f}(\mathbf{x}_i)$, ex.

$$f_1 = x_1^2, f_2 = x_2^2, f_3 = \sqrt{2}x_1x_2$$

- Nous avons un « noyau » de $K(\mathbf{x}_i,\mathbf{x}_j)= (\mathbf{x}_i \cdot \mathbf{x}_j)^2$
- Généralement: On peut trouver de façon efficace des séparateur linéaires optimaux dans des espaces des caractéristiques de dimension de l'ordre du milliard, ou dans certains cas infinie
- Important : nous avons un séparateur non-linéaire dans l'espace original (c.-à-d. **x**)

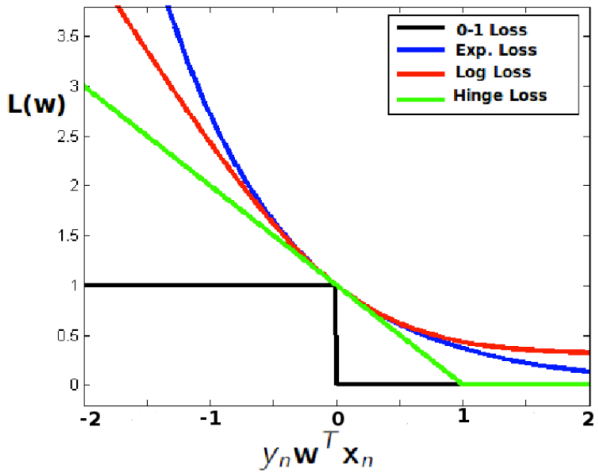
L'astuce de noyau (RBF) et la régression logistique

$$\max\left[\sum_{i=1}^{N}\left(\mathbf{y}_i^T\theta\mathbf{f}(\mathbf{x}_i)-\log\left\{\sum_i\exp\left(\mathbf{y}_i^T\theta\mathbf{f}(\mathbf{x}_i)\right)\right\}\right)\right]$$

Comparez avec la même idée appliqué aux MVS

$$\min_{\mathbf{w},b}\ \|\mathbf{w}\|\ ,\quad y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq C(1-\xi_i),\quad \forall \mathrm{i}$$

Comparez des « fonctions de perte » :

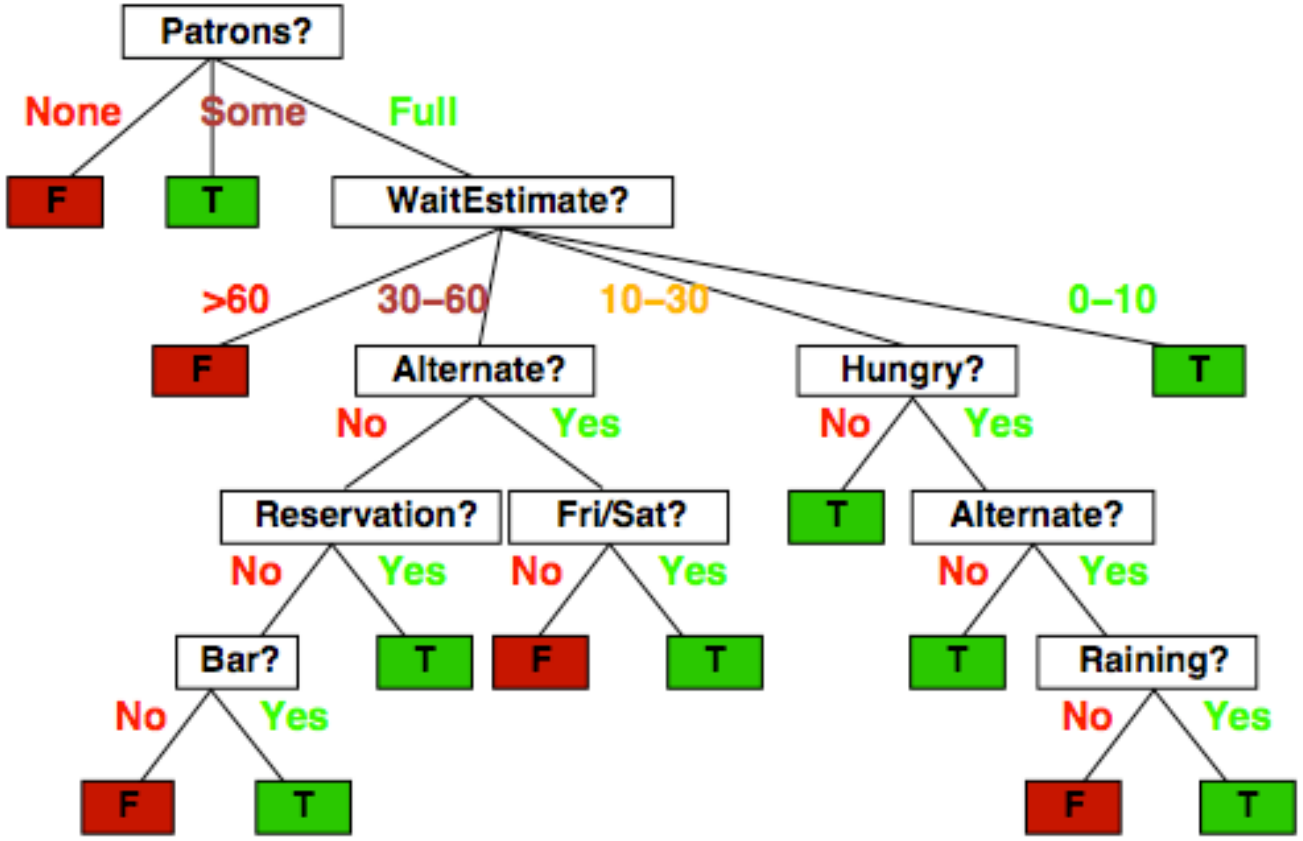


0-1	$1-\mathrm{H}(0)$, (1-Heaviside)
hinge	$1-y_i\mathbf{w}^T\mathbf{x}_i,\ \max\{0,1-y_i\mathbf{w}^T\mathbf{x}_i\}$
log	$\log(1+\exp(y_i\mathbf{w}^T\mathbf{x}_i))$
exp	$\exp(-y_i\mathbf{w}^T\mathbf{x}_i)$

Les arbres de décision

et apprentissage

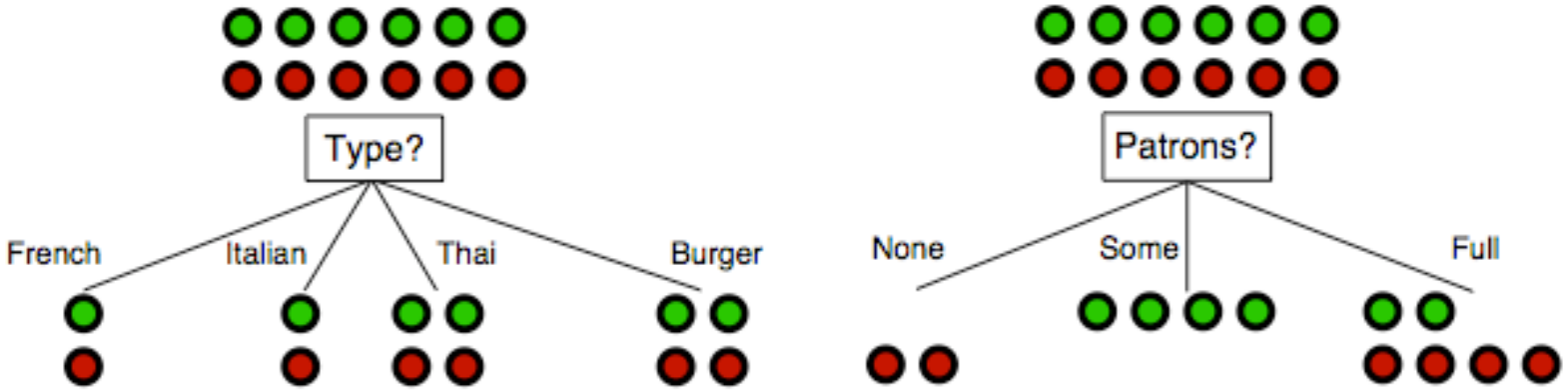
Arbre de décision pour décider d'attendre ou non une table



Exemple: Ensemble d'exemples

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Répartition des exemples en testant les attributs



- Le choix de type n'aide pas à différentier les exemples négatifs des positifs
- Le choix de Clients parvient bien à séparer les exemples

L'entropie

- Dans la théorie de l'information il existe le concept d'entropie
- L'entropie mesure le montant d'incertitude

$$H(P(v_1),\ldots P(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

- si les réponses possible v_i ont les probabilités $P(v_i)$, exemple: une pièce de monnaie

$$H(P(0),P(1)) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

Comment choisir l'attribut

- En utilisant la théorie de l'information
- Soit *N* le nombre d'exemples
- Soit une variable de décision d ayant les valeurs possibles $d_1, d_2 \ldots d_k$
- Supposons que l'ensemble d'exemples contient n_i items pour chaque valeur possible d_i
- Initialement la valeur de l'entropie est

$$Entropie_I = \sum_{i=1}^{k} -\left(\frac{n_i}{N}\right)\log_n\left(\frac{n_i}{N}\right)$$

Comment choisir l'attribut (suite)

- Supposons maintenant que l'on choisisse l'attribut A, qui a les valeurs possibles $a_1, a_2 \ldots a_j$
- Si on choisit la valeur a_i, on se retrouve avec un sous-ensemble de M_i exemples tel que on a m_v items pour chaque valeur de décision $v = d_1 \ldots d_k$
- L'entropie dans ce cas est donc $$E(a_i) = \sum_{v=1}^{k} -\left(\frac{m_v}{M_i}\right)\log_2\left(\frac{m_v}{M_i}\right)$$
- On fait la moyenne sur toutes les valeurs de A : $$\text{Reste}(A) = \sum_{i=1}^{j}\left(\frac{M_i}{N}\right)E(a_i)$$
- On choisit alors l'attribut A qui maximise le gain $$\text{Gain}(A) = \text{Entropie}_I - \text{Reste}(A)$$

Apprentissage d'arbre de décision

fonction CREER-ARBRE-DECISION(*exemples, attributs, défaut*)
si *exemples* = $\varnothing$ **retourner** *défaut*
sinon si tous les exemples ont la même classification **alors**
retourner cette classification
sinon si attributs = $\varnothing$ **alors**
retourner la valeur majoritaire parmi les exemples
sinon
meilleur **<-** CHOISIR-ATTRIBUT*(attributs, exemples*)
arbre **<-** nouvel arbre avec attribut *meilleur* comme racine
m **<-** la valeur majoritaire parmi les exemples
pour chaque valeur v_i de *meilleur* **faire**
$exemples_i$ **<-** { e $\in$ *exemples* | valeur de *meilleur* pour $e = v_i$ }
attributs' = *attributs* – {*meilleur* }
sous-arbre = CREER-ARBRE-DECISION($exemples_i$,*attributs', m*)
ajouter une branche à *arbre* avec attribut v_i et *sous-arbre*
retourner *arbre*

Arbre de décision - exemple

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Entropie = - 0,5*log(0,5) – 0,5*log(0,5)
= 1

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: - 0,66 log 0,66 – 0,33 log 0,33

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92
Entropie si A = 1: - 0,33 log 0,33 – 0,66 log 0,66

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92
Entropie si A = 1: 0,92

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92
Entropie si A = 1: 0,92
Entropie si A = 2:

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92
Entropie si A = 1: 0,92
Entropie si A = 2: - 0,5 log 0,5 – 0,5 log 0,5

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92
Entropie si A = 1: 0,92
Entropie si A = 2: 1

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0,92
Entropie si A = 1: 0,92
Entropie si A = 2: 1
Moyenne = 0,3*0,92 + 0,3*0,92 + 0,4*1
= 0,95
Gain = 1 – 0,95 = 0,05

Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut B:
Entropie si B = 0: 0,92
Entropie si B = 1: 1
Entropie si B = 2: 0,97
Moyenne = 0,3*0,92 + 0,2*1 + 0,5*0,97
= 0,96
Gain = 1 – 0,96 = 0,04

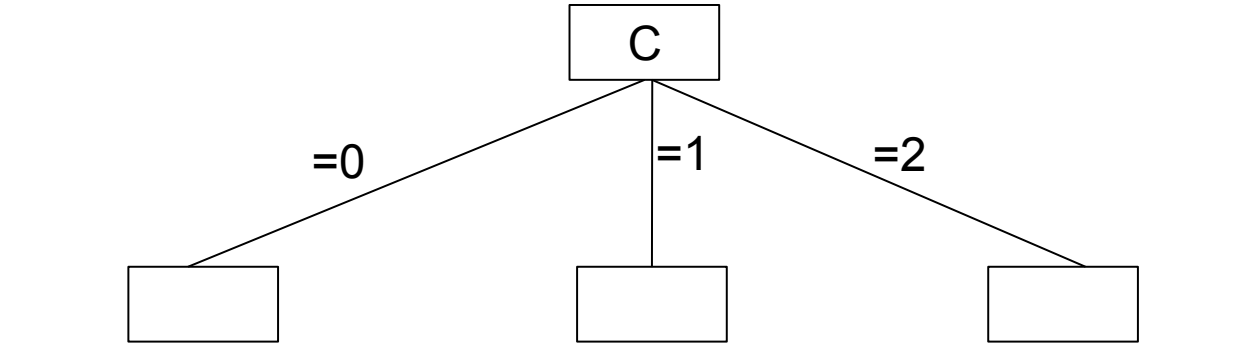
Arbre de décision – exemple (suite)

A	*B*	*C*	*Décision*
0	2	1	1
1	1	0	1
2	2	1	1
2	0	2	1
0	0	0	1
1	2	2	0
0	2	2	0
1	0	0	0
2	1	2	0
2	2	2	0

Attribut C:
Entropie si C = 0: 0,92
Entropie si C = 1: 0
Entropie si C = 2: 0,72
Moyenne = 0,3*0,92 + 0,2*0 + 0,5*0,72
= 0,64
Gain = 1 – 0,64 = 0,36

On choisit donc l'attribut C comme racine de l'arbre

Arbre de décision (partiel)



A	*B*	*C*	*Décision*
1	1	0	1
0	0	0	1
1	0	0	0

A	*B*	*C*	*Décision*
0	2	1	1
2	2	1	1

A	*B*	*C*	*Décision*
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

Arbre de décision – exemple (suite)

Cas C = 0

A	*B*	*C*	*Décision*
1	1	0	1
0	0	0	1
1	0	0	0

Entropie initiale : 0,92

Arbre de décision – exemple (suite)

Cas C = 0

A	*B*	*C*	*Décision*
1	1	0	1
0	0	0	1
1	0	0	0

Attribut A:
Entropie si A = 0: 0
Entropie si A = 1: 1
Moyenne = 0,33*0 + 0,66*1
= 0,66
Gain = 0,92 – 0,66 = 0,26

Arbre de décision – exemple (suite)

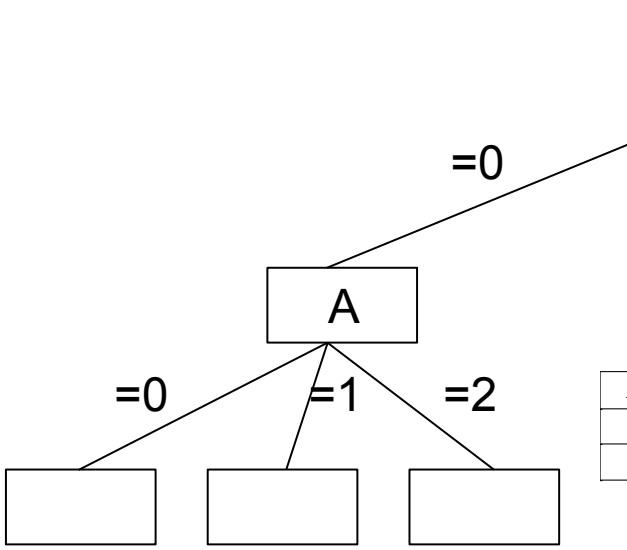
Cas C = 0

A	*B*	*C*	*Décision*
1	1	0	1
0	0	0	1
1	0	0	0

Attribut B:
Entropie si B = 0: 1
Entropie si B = 1: 0
Moyenne = 0,66*1 + 0,33*0
= 0,66
Gain = 0,92 – 0,66 = 0,26

Les deux attributs sont équivalents. Pas d'importance que l'on choisisse A ou B.

Arbre de décision (partiel)



A	B	C	Décision
0	2	1	1
2	2	1	1

A	B	C	Décision
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

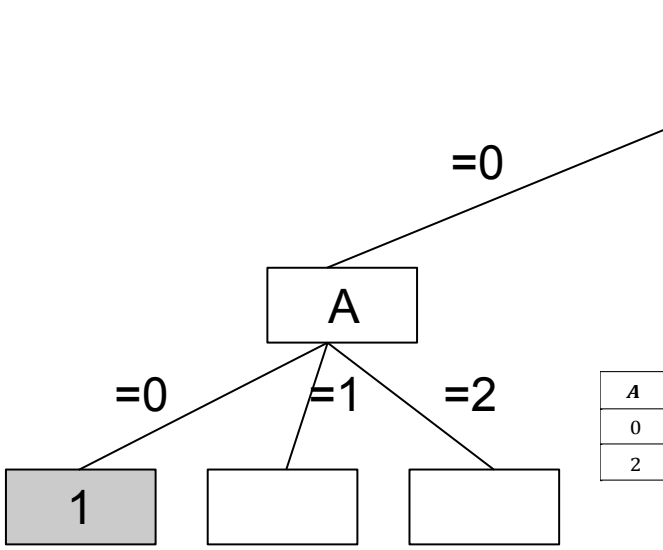
A	B	C	Décision
0	0	0	1

A	B	C	Décision
1	1	0	1
1	0	0	0

Arbre de décision – exemple (suite)

Cas C = 0 et A = 0
Tous les exemples ont la même valeur.
On retourne donc cette valeur.

Arbre de décision (partiel)



A	B	C	Décision
1	1	0	1
1	0	0	0

A	B	C	Décision
0	2	1	1
2	2	1	1

A	B	C	Décision
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

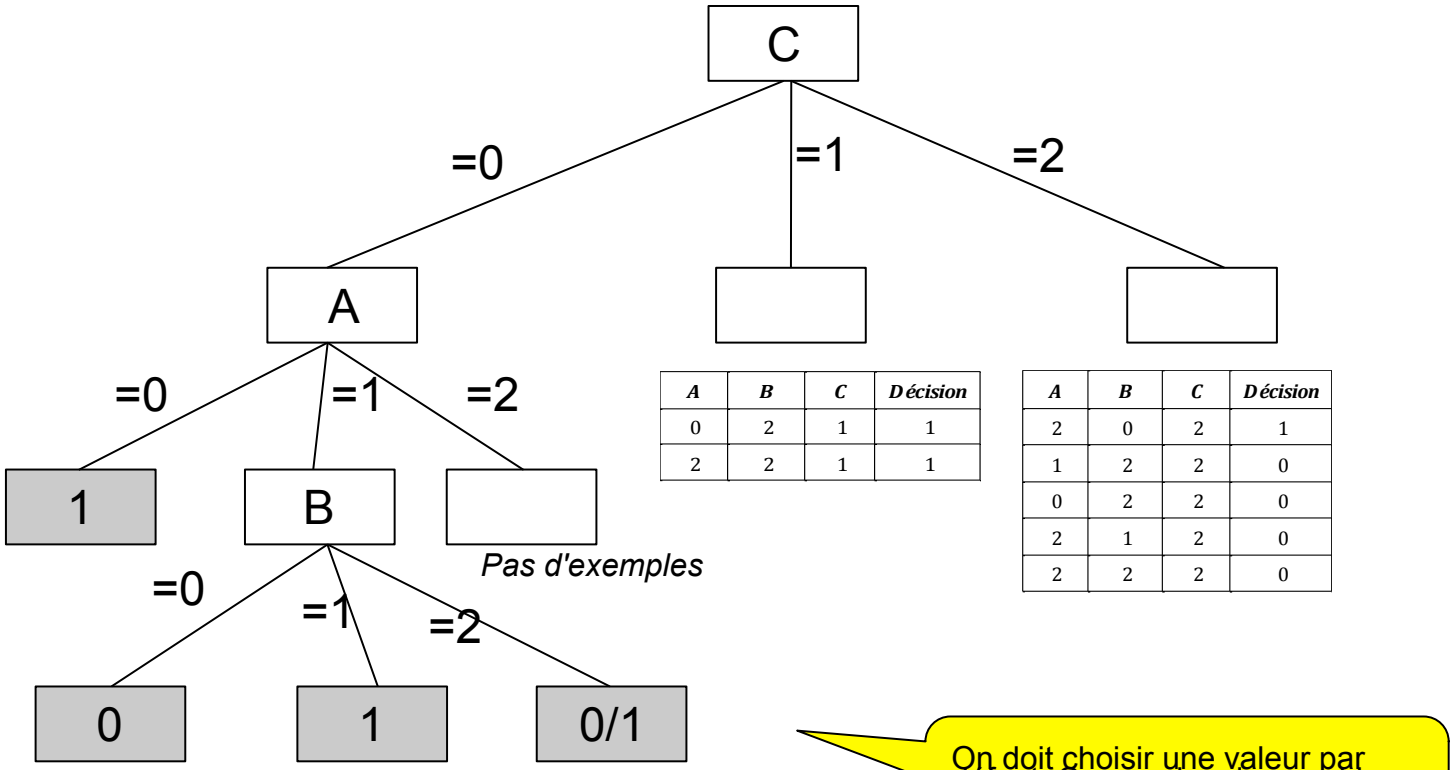
Arbre de décision – exemple (suite)

Cas C = 0 et A = 1

Pour B = 0 et B = 1, tous les exemples sont de la même classe (un seul exemple pour chaque cas)

Pour B = 2, on a aucun exemple. On doit donc choisir la valeur majoritaire au noeud parent. Comme il n'y a que deux exemples et qu'on a 1 dans un cas et 0 dans l'autre, on peut choisir indifféremment l'une ou l'autre.

Arbre de décision (partiel)



A	B	C	Décision
0	2	1	1
2	2	1	1

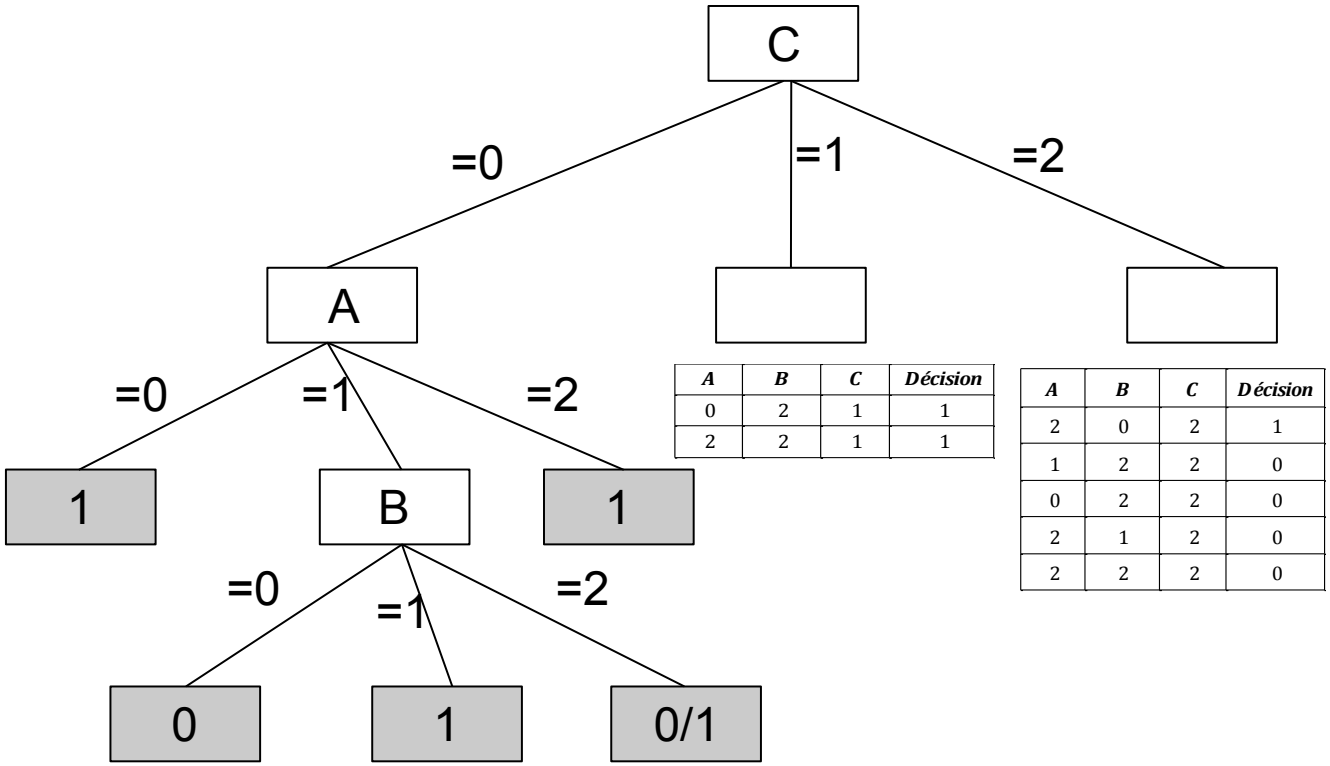
A	B	C	Décision
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

On doit choisir une valeur par défaut. Comme les deux ont les mêmes proportions, on peut prendre l'une ou l'autre

Arbre de décision – exemple (suite)

Cas C = 0 et A = 2
Aucun exemple.
Pour C = 0 la valeur majoritaire est 1
On retourne donc cette valeur par défaut:

Arbre de décision (partiel)



A	*B*	*C*	*Décision*
0	2	1	1
2	2	1	1

A	*B*	*C*	*Décision*
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

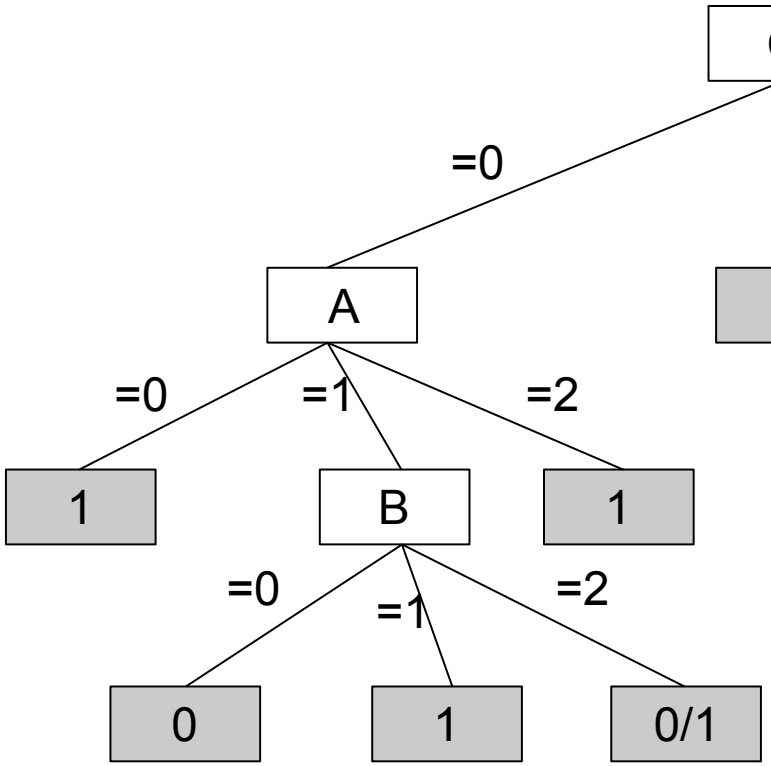
Arbre de décision – exemple (suite)

Cas C = 1

A	*B*	*C*	*Décision*
0	2	1	1
2	2	1	1

Tous les exemples sont de la même catégorie.

Arbre de décision (partiel)



A	*B*	*C*	*Décision*
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

Arbre de décision – exemple (suite)

Cas C = 2

A	*B*	*C*	*Décision*
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

Entropie initiale : 0,72

Arbre de décision – exemple (suite)

Cas C = 2

A	*B*	*C*	*Décision*
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

Attribut A:
Entropie si A = 0: 0
Entropie si A = 1: 0
Entropie si A = 2: 0,92
Moyenne = 0,2*0 + 0,2*0 + 0,6*0,92
= 0,55
Gain = 0,72 – 0,55 = 0,17

Arbre de décision – exemple (suite)

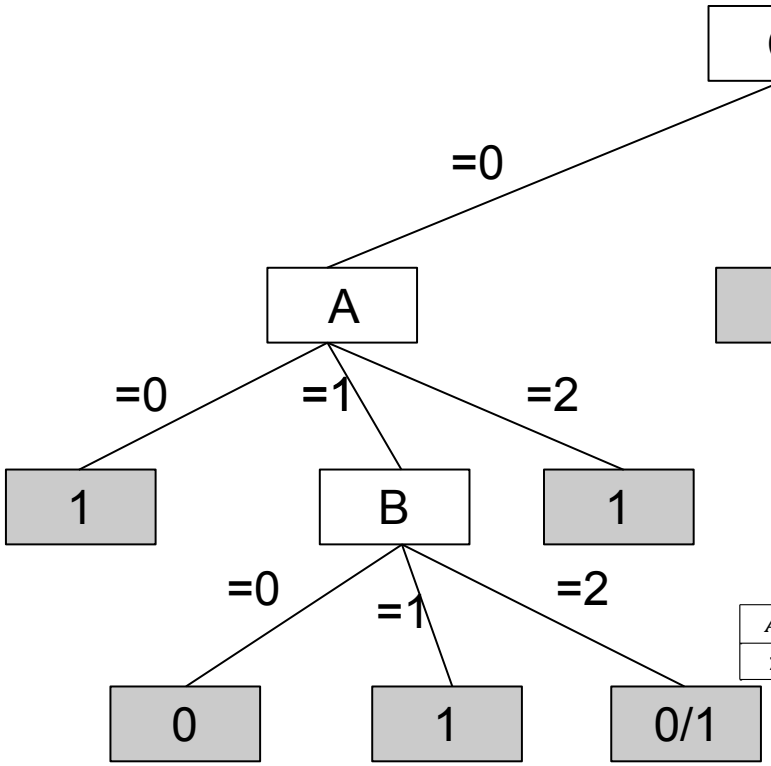
Cas C = 2

A	*B*	*C*	*Décision*
2	0	2	1
1	2	2	0
0	2	2	0
2	1	2	0
2	2	2	0

Attribut B:
Entropie si B = 0: 0
Entropie si B = 1: 0
Entropie si B = 2: 0
Moyenne = 0,2*0 + 0,2*0 + 0,6*0
= 0
Gain = 0,72 – 0 = 0,72

On choisit l'attribut B

Arbre de décision (partiel)

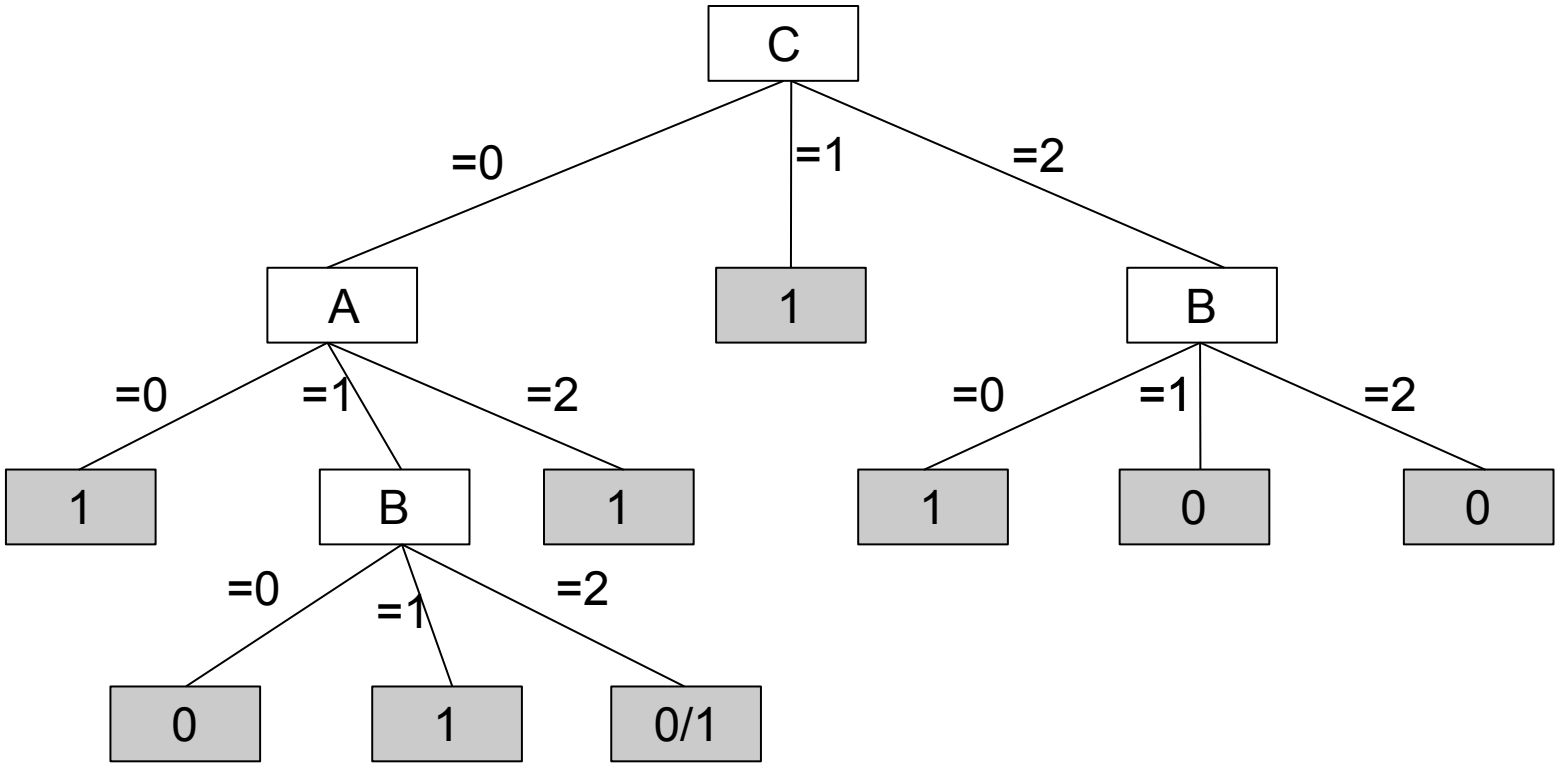


A	B	C	Décision
2	0	2	1

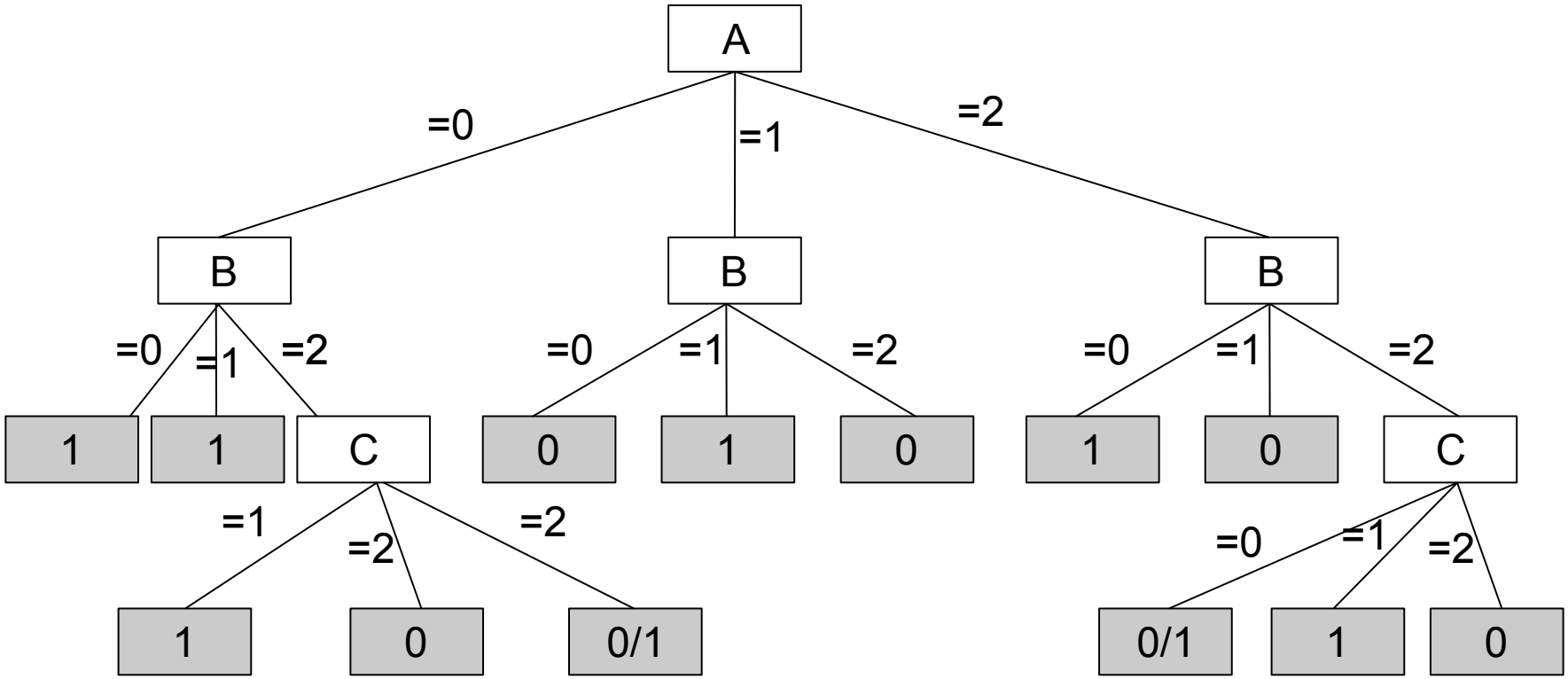
A	B	C	Décision
2	1	2	0

A	B	C	Décision
1	2	2	0
0	2	2	0
2	2	2	0

Arbre de décision final



Arbre de décision avec ordre A,B,C



On obtient un arbre plus complexe

Méthodologie d'apprentissage

- Collecter un grand ensemble d'exemples
- Diviser en un ensemble d'entraînement et un ensemble de test
- Appliquer l'algorithme sur l'ensemble d'entraînement afin de générer l'hypothèse *h*
- Mesurer la proportion d'exemples de l'ensemble test qui sont correctement classés par l'hypothèse *h*
- Répéter avec d'autre partitions de l'ensemble d'exemples

Ratio de gain

- Problème: un attribut à plusieurs valeurs peut présenter un meilleur gain, par le simple fait de la tendance à répartir les exemples dans de petits ensembles
- Solution: diviser le gain par la valeur d'entropie de cet attribut:

$$\sum_{i=1}^{k} -(n_i / N)\ log_2(n_i / N)$$

où k est le nombre de valeurs pour l'attribut en question, N le nombre total d'exemples pour ce noeud, et n_i le nombre d'occurrences de la $i$ème valeur de l'attribut

Quand stopper la subdivision?

- Première solution:
  - À chaque niveau de subdivision dans l'arbre, on teste avec l'ensemble de validation
  - Lorsque l'erreur atteint un minimum, on arrête
- Deuxième solution:
  - On fixe un seuil minimal de gain
  - Difficile de fixer cette valeur
- Troisième solution:
  - On arrête la subdivision d'un noeud lorsque le nombre d'exemples tombe en dessous d'un certain seuil (nombre absolu ou % de l'ensemble initial)

Quand stopper la subdivision? (suite)

- Quatrième solution:
  - Condition d'arrêt basée sur un critère global:

$$\alpha \cdot \text{taille} + \sum_{N \in \text{feuilles}} \text{Entropie}(N)$$

  - Ici la taille peut être le nombre total de noeuds ou de branches
  - Difficile de fixer α

Quand stopper la subdivision? (suite)

- Cinquième solution:
  - Utilisation d'un test d'hypothèse pour vérifier si le gain est significatif
  - Test de chi-carré
- Sixième solution:
  - Élagage
  - On produit tout l'arbre
  - Toutes les feuilles qui sont enfants d'un même noeud sont éliminées si elles apportent peu de gain

Apprentissage par ensembles

- On combine un ensemble d'hypothèses
- Chaque hypothèse vote et les votes sont pondérés
- Technique couramment utilisée: dopage
- L'algorithme le plus populaire: Adaboost
- Propriété importante de Adaboost: si on l'applique à un algorithme d'apprentissage faible, il retourne une hypothèse qui classe parfaitement les données d'apprentissage si le nombre d'hypothèses dans l'ensemble est grand

ADA-BOOST

fonction ADA-BOOST(*exemples,L,M*) **retourne** hypothèse pondérée
  entrées: *exemples,* ensemble de N exemples
  L, algorithme d'apprentissage
  M, nombre d'hypothèses
  w est un vecteur de *N* poids, initalement fixés à *1/N*
  h est un vecteur de *M* hypothèses
  z est un vecteur de *M* poids
  pour *m = 1* **à** *M* **faire:**
    h[*m*] = L(*exemples*, **w**)
    erreur = 0
    pour *j = 1* **à** *N* **faire:**
      si h[*m*](x_j) ≠ y_i **alors** *erreur* = *erreur* + **w**[*j*]
    pour *j = 1* **à** *N* **faire:**
      si h[*m*](x_j) = y_i **alors w**[*j*] = **w**[*j*] × *erreur* / (1 – *erreur*)
    w = NORMALISER(**w**)
    z = log(1 – *erreur*) / *erreur*
  retourner MAJORITE_PONDEREE(**h**,**z**)

More Advanced Boltzmann Machines and 'Harmonium' Based Methods

Chris Pal

Overview

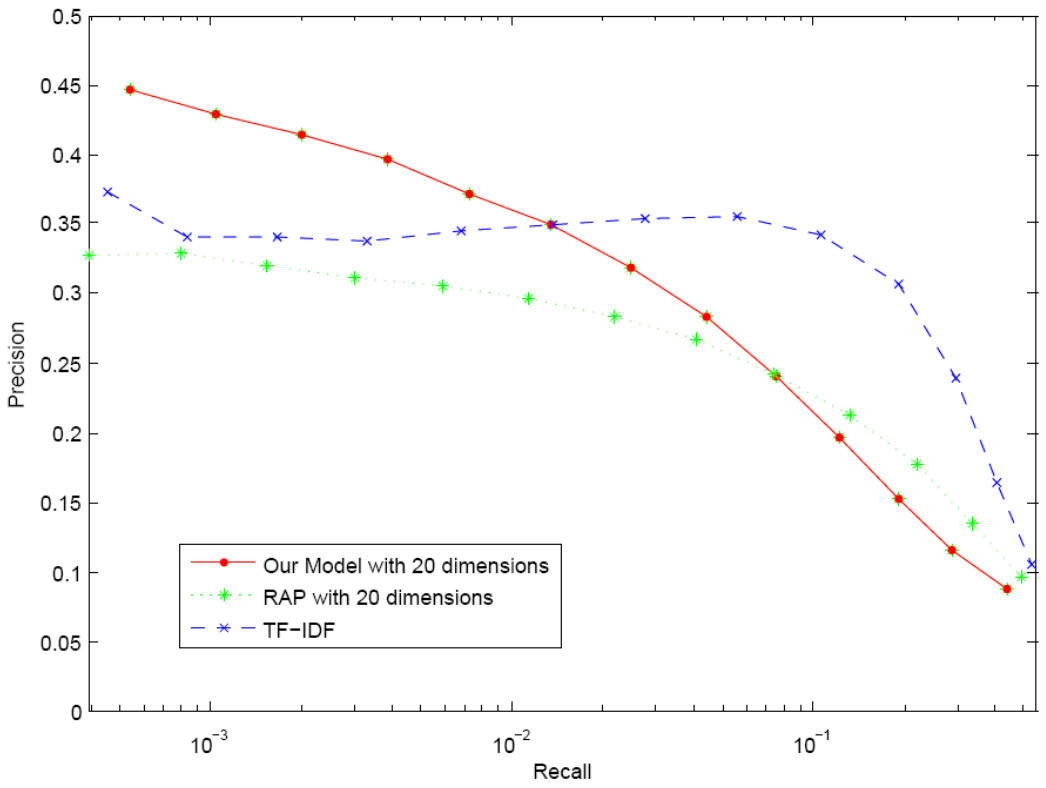
- New Topic Models, Start Simple & Build
  - Compare with related model structures
  - Precision vs. Recall 20 Newsgroups

- Add Authors + Discriminative Methods
  - Predict NIPS Authors & Email Recipients

- Authors + Recipients & Creating (DARTs)
  - Transfer Learning in Social Networks
  - Experiments with Enron Email

Our Model (MCA) vs. TFIDF vs. RAP

MRR	Method
.45	Our Model
.37	TFIDF
.33	RAP



- Precision vs. Recall on 20 Newsgroups, 100 word vocabulary
- 20 dimensional hidden topic space
- Cosine Distance Comparisons (.9, .1 – Train, Test Split)
- Compared with TFIDF and Rate Adapting Poisson (RAP) Model

20 Newsgroups

- 10,000 word vocab. - highest MI with class
- 18,796 documents
- Downcased, no stopwords, porter stemmed
- comp., rec., sci., .forsale, .politics, .religion

NIPS

- 13,649 word vocab.
- 1,740 papers
- Downcased, no stopwords, no stemming
- 13 years of NIPS proceedings 1987-1999

20 Newsgroups Topics

Religon				Images				Computer			
god	.0107	max	-.0256	jpeg	.0125	god	-.0051	window	.0130	god	-.0141
peopl	.0074	giz	-.0183	gif	.0077	wire	-.0033	graphic	.0115	jpeg	-.0073
lord	.0053	bhj	-.0179	imag	.0064	lord	-.0026	pub	.0110	jehovah	-.0073
jehovah	.0052	output	-.0171	color	.0053	law	-.0024	server	.0109	lord	-.0064
armenian	.0051	qax	-.0156	qualiti	.0045	presid	-.0024	ftp	.0108	jesu	-.0053
jesu	.0046	entri	-.0144	format	.0042	jesu	-.0024	system	.0100	peopl	-.0043
presid	.0045	bxn	-.0130	viewer	.0042	entri	-.0023	mail	.0096	christ	-.0041
don	.0036	file	-.0119	compress	.0037	jehovah	-.0021	data	.0090	father	-.0033
christian	.0036	program	-.0115	convert	.0035	state	-.0021	user	.0084	christian	-.0030
live	.0033	window	-.0090	displai	.0033	christian	-.0020	comput	.0081	don	-.0029
christ	.0032	nrhj	-.0085	pixel	.0032	year	-.0020	anonym	.0080	armenian	-.0026
govern	.0031	line	-.0084	quantiz	.0029	question	-.0019	softwar	.0078	son	-.0022
dai	.0031	biz	-.0077	free	.0028	live	-.0019	widget	.0078	mormon	-.0018
didn	.0030	printf	-.0072	graphic	.0028	ground	-.0018	applic	.0077	bibl	-.0014
father	.0030	check	-.0070	bit	.0027	christ	-.0018	list	.0076	output	-.0014
state	.0028	jpeg	-.0068	version	.0027	armenian	-.0018	includ	.0074	vers	-.0014
thing	.0027	char	-.0067	zip	.0025	dai	-.0018	support	.0072	gif	-.0013
law	.0026	stream	-.0066	softwar	.0024	hous	-.0016	run	.0072	didn	-.0013
made	.0024	section	-.0066	quicktim	.0024	person	-.0016	version	.0071	sin	-.0013
fact	.0022	info	-.0064	mirror	.0024	time	-.0016	motif	.0070	live	-.0012

NIPS Topics

Biological Neuroscience				Reinforcement Learning				Probabilistic Methods			
cells	.439	training	-.556	learning	.318	image	-.536	data	.364	state	-.512
cell	.361	networks	-.500	policy	.266	data	-.444	model	.307	time	-.454
firing	.360	error	-.472	reinforcement	.252	images	-.431	mixture	.271	neuron	-.449
cortex	.357	network	-.470	control	.239	recognition	-.345	gaussian	.260	neural	-.429
cortical	.355	speech	-.465	state	.234	feature	-.315	likelihood	.225	system	-.422
stimulus	.327	neural	-.461	action	.233	object	-.271	image	.221	control	-.405
spike	.314	classifier	-.436	actions	.158	visual	-.270	distribution	.217	neurons	-.373
synaptic	.310	class	-.412	weight	.153	features	-.263	bayesian	.213	analog	-.363
synapses	.275	word	-.410	states	.151	gaussian	-.241	images	.204	network	-.359
motion	.268	state	-.407	controller	.150	classification	-.233	em	.189	circuit	-.335
orientation	.262	recognition	-.406	optimal	.125	mixture	-.227	density	.183	action	-.334
excitatory	.255	classifiers	-.386	weights	.121	models	-.217	models	.182	synaptic	-.317
visual	.253	classification	-.370	error	.117	model	-.211	posterior	.163	chip	-.316
inhibitory	.243	set	-.359	time	.115	likelihood	-.190	prior	.148	networks	-.287
response	.243	hmm	-.354	neuron	.105	set	-.189	regression	.146	states	-.285
stimuli	.240	algorithm	-.344	sutton	.102	orientation	-.184	kernel	.144	memory	-.279
spatial	.238	hidden	-.342	gradient	.101	classifier	-.180	log	.135	recurrent	-.263
direction	.233	test	-.337	recurrent	.101	face	-.179	classification	.134	current	-.263
membrane	.231	mixture	-.334	agent	.096	class	-.171	class	.133	policy	-.259
eye	.229	data	-.333	learn	.096	test	-.169	parameters	.124	reinforcement	-.256

Multi-Conditional Learning

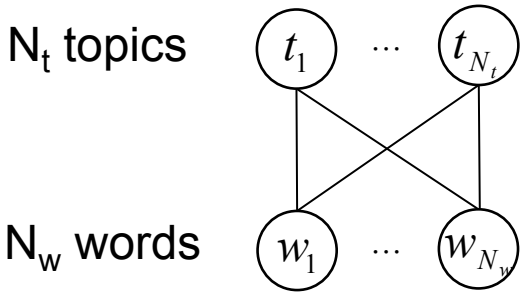
- One motivation:
  - Conditional Random Fields can be derived from a traditional joint model
  - But, there are many other conditional distributions that could be defined
- What do we gain if we model those as well?

$$\begin{aligned}\mathcal{L}_{MC} &= \log\left(P(\mathbf{y}|\mathbf{x})^{\alpha} P(\mathbf{x}|\mathbf{y})^{\beta}\right) \\ &= \alpha\mathcal{L}_{y|x}(\boldsymbol{\theta}) + \beta\mathcal{L}_{x|y}(\boldsymbol{\theta}).\end{aligned}$$

- Other combinations possible

$$\mathcal{L} = \log(P(\mathbf{y}|\mathbf{x})^{\alpha} P(\mathbf{x})^{\beta})$$

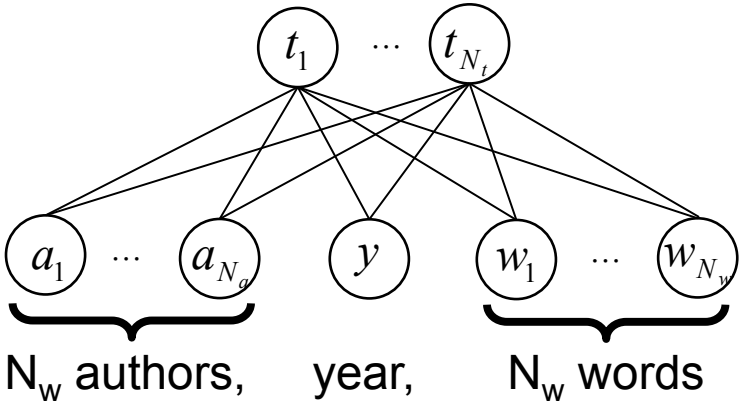
Discriminative Training, MCL and a Richer Model



Maximum Likelihood $P(\{w\})$

Discriminative Training $P(w_i \mid w_{j:j\neq i})$

'Multi-conditional' Training $\prod_i P(w_i \mid w_{j:j\neq i})$



A Richer Model

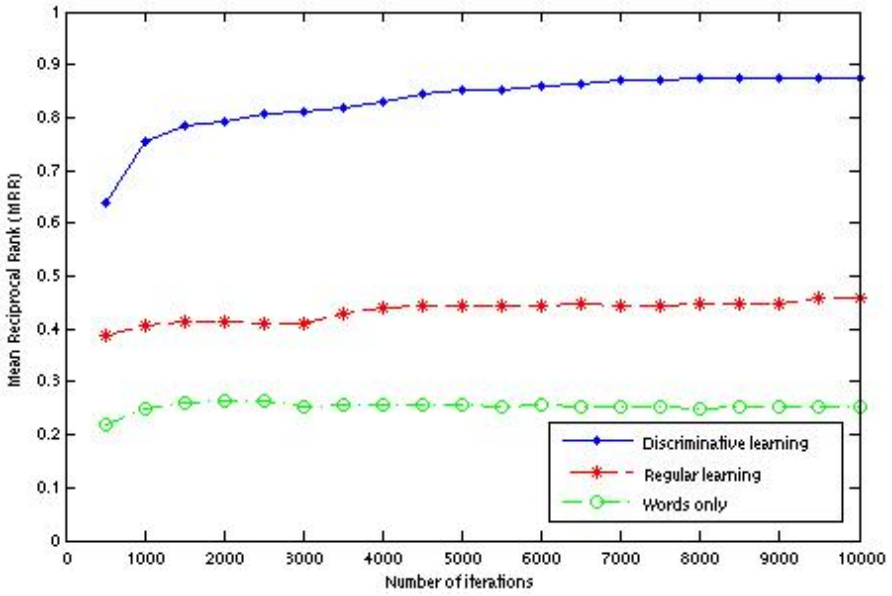
Discriminative Training $P(\{a\} \mid \{w\}, y)$

NIPS Topics
Multi-conditional Learning

Pattern Recognition				'Neural Networks'				Classification and Regression			
recognition	.734	policy	-.574	input	.900	itly	-.106	functions	.419	units	-.075
image	.687	weight	-.537	output	.820	construc	-.105	class	.403	visual	-.069
images	.663	action	-.504	hidden	.617	nash	-.100	classifier	.391	motion	-.068
object	.577	reinforcement	-.454	model	.593	ination	-.099	regression	.368	unit	-.057
speech	.547	learning	-.428	state	.550	probabilit	-.098	classifiers	.361	task	-.055
visual	.489	convergence	-.420	speech	.544	rival	-.097	bounds	.350	direction	-.054
word	.483	optimal	-.414	training	.540	aleksander	-.097	gaussian	.330	learning	-.052
features	.465	actions	-.400	models	.538	laxation	-.097	loss	.323	eye	-.050
feature	.419	error	-.395	weights	.529	arthur	-.096	theorem	.322	object	-.049
objects	.384	neuron	-.344	error	.506	duplicating	-.096	density	.317	motor	-.048
face	.379	controller	-.341	patterns	.504	cedures	-.096	approximation	.315	cortex	-.048
hmm	.287	gradient	-.338	inputs	.502	affirmative	-.095	bound	.315	action	-.047
classification	.284	theorem	-.335	unit	.500	hindered	-.095	matrix	.314	velocity	-.047
segmentation	.272	reward	-.330	weight	.500	allan	-.094	classification	.312	position	-.044
system	.262	sutton	-.328	net	.489	glasgow	-.094	vector	.312	activity	-.044
context	.261	finite	-.324	architecture	.488	delaying	-.094	kernel	.302	control	-.042
frame	.258	function	-.308	word	.483	mutations	-.094	distribution	.298	cortical	-.042
classifier	.257	stochastic	-.304	systems	.482	mech	-.094	log	.290	reinforcement	-.041
orientation	.255	control	-.296	control	.474	concomitantly	-.093	data	.277	head	-.041
vision	.245	time	-.291	learning	.471	mother	-.093	neural	.270	cells	-.040

Optimize an objective based on the product of the conditional probability for one word given all the others.

Predicting NIPS Authors



MRR	Method
.88	Discriminative
.46	Joint
.25	Joint, Words only

- Comparing Models, Mean Reciprocal Rank (MRR)
- Cosine Distance Comparisons (.9, .1 – Train, Test Split)

NIPS Topics + Authors

"VLSI Topic"									
Words				**People**				**Year**	
analog	0.080	basis	-0.091	Platt, J	0.207	Sejnowski, T	-0.466	1991	0.138
pulse	0.068	representation	-0.089	Harris, J	0.205	Mel, B	-0.384	1997	0.122
chip	0.066	pca	-0.086	Alspector, J	0.153	Dayan, P	-0.349	1992	0.080
vlsi	0.066	representations	-0.084	Koch, C	0.153	Mozer, M	-0.346	1993	0.051
velocity	0.056	class	-0.084	Lazzaro, J	0.146	Zemel, R	-0.331	1998	0.050
synapse	0.049	structure	-0.081	Principe, J	0.137	Cottrell, G	-0.314	1989	0.032
circuit	0.047	face	-0.079	Mead, C	0.135	Tenenbaum, J	-0.304	1987	0.028
voltage	0.042	mixture	-0.078	Cauwenberghs, G	0.112	Wiles, J	-0.293	1996	0.021
trajectory	0.042	context	-0.075	Mjolsness, E	0.108	Pouget, A	-0.283	1999	0.000
circuits	0.041	zemel	-0.072	Maass, W	0.100	Ahmad, S	-0.267	1990	-0.013
"Vision Science Topic"									
Words				**People**				**Year**	
orientation	0.083	decision	-0.095	Sejnowski, T	0.398	Bartlett, P	-0.278	1989	0.038
dominance	0.080	bounds	-0.089	Koch, C	0.375	Jaakkola, T	-0.240	1991	0.032
visual	0.080	risk	-0.088	Pouget, A	0.289	Shavlik, J	-0.239	1988	0.029
ocular	0.079	theorem	-0.087	Kawato, M	0.263	Tesauro, G	-0.207	1994	0.006
velocity	0.078	margin	-0.080	Obermayer, K	0.260	Thrun, S	-0.207	1996	0.005
stimuli	0.075	trees	-0.075	Nowlan, S	0.230	Bengio, Y	-0.199	1997	0.002
eye	0.075	boosting	-0.072	Dayan, P	0.219	Kowalczyk, A	-0.194	1999	0.000
cortex	0.070	policy	-0.070	Zemel, R	0.214	Cohn, D	-0.181	1992	-0.013
cortical	0.069	cost	-0.069	Lee, D	0.199	Baluja, S	-0.178	1990	-0.038
lgn	0.068	algorithms	-0.067	Li, Z	0.198	Lee, Y	-0.177	1987	-0.048

NIPS Papers (For Context)

Terrence J. Sejnowski —*Recurrent Eye Tracking Network Using a Distributed Representation of Image Motion.* NIPS 1991 —*Combining Visual and Acoustic Speech Signals with a Neural Network Improves Intelligibility.* NIPS 1989
John C. Platt — *Analog Circuits for Constrained Optimization.* NIPS 1989 — *An Analog VLSI Chip for Radial Basis Functions.* NIPS 1992
Christof Koch — *An Integrated Vision Sensor for the Computation of Optical Flow Singular Points.* NIPS 1998 — *Analog VLSI Circuits for Attention-Based, Visual Tracking.* NIPS 1996 — *An Analog VLSI Saccadic Eye Movement System.* NIPS 1993

Academic Email

- 4,643 emails
- 190 recipients
- 8,693 word vocabulary
- Downcased, no stopwords, no stemming

Mean Reciprocal Rank (MRR) Evaluation

Reciprocal of the rank at which the first relevant response was returned

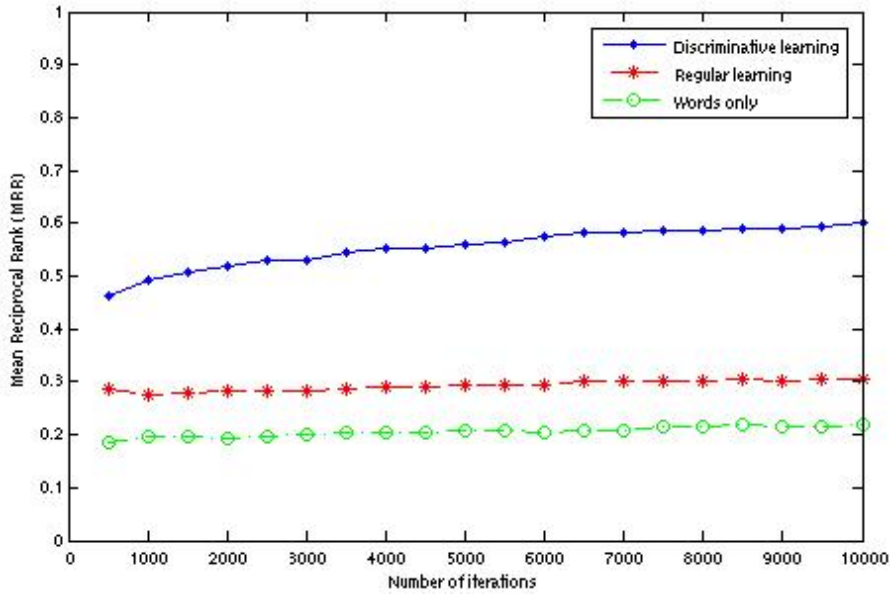
Method 1: Use the cosine of all previous sent email,
obtain authors from ordered closest match

Method 2: Use model to make predictions
obtain ordered list from probability distribution

Academic Email Topics

"Writing a Paper on Coreference"									
Words				**People**				**Month**	
paper	0.053	email	-0.019	fuchun	0.412	culotta	-0.073	5	0.470
model	0.047	people	-0.015	wellner	0.395	ronb	-0.065	2	0.335
section	0.037	find	-0.003	mhay	0.373	traustik	-0.030	9	0.208
inference	0.034	addrie	-0.001	jensen	0.192	viola	-0.021	6	0.157
results	0.030	clyde	-0.001	pereira	0.085	system	-0.021	3	0.106
models	0.028	calobase	-0.000	lafferty	0.073	lsaul	-0.017	8	0.031
work	0.028	sgml	-0.000	kate	0.060	souccar	-0.013	7	0.012
coreference	0.025	green	-0.000	mahadeva	0.050	tzhang	-0.013	10	0.000
ben	0.024	remotely	-0.000	casutton	0.045	jst@	-0.012	4	-0.135
text	0.023	emacs	-0.000	jean	0.040	szmiller	-0.012	1	-0.223
"Building a Contact Finding System"									
Words				**People**				**Month**	
aron	0.026	model	-0.006	culotta	0.357	fuchun	-0.168	1	0.334
data	0.021	research	-0.006	weili	0.116	wellner	-0.153	2	0.309
email	0.018	inference	-0.003	ronb	0.111	mhay	-0.131	9	0.112
people	0.015	spider	-0.003	pereira	0.105	saunders	-0.109	7	0.012
find	0.011	paper	-0.003	viola	0.097	mikem	-0.062	6	0.003
results	0.011	mccallum	-0.003	lafferty	0.086	jensen	-0.041	10	0.000
ron	0.010	fuchun	-0.002	traustik	0.074	adingle	-0.035	4	-0.060
calo	0.010	papers	-0.002	ghuang	0.046	slfeng	-0.015	5	-0.064
class	0.009	prototype	-0.002	hough	0.032	jean	-0.014	3	-0.093
training	0.009	mysql	-0.002	casutton	0.031	msindela	-0.014	8	-0.176

Predicting Email Recipients



MRR	Method
.60	Discriminative
.30	Joint
.21	Joint, Words only

- Comparing Models, Mean Reciprocal Rank (MRR)
- Cosine Distance Comparisons (.9, .1 – Train, Test Split)
- 20 dimensional hidden ‘topic’ space

Summary of Results (so far)

Data Set	MCA words only	PCA with authors	MCA reg. learning	MCA discrim. learning
NIPS	.26	.40	.45	**.87**
Email	.22	.27	.30	**.60**

- Richer model with authors included helps
- Discriminative optimization helps a lot

Inference approximatif

Méthodes d'inférence approximative

- Propagation de croyance dans un graphe avec un cycle
- Coupes de graphe
- Méthodes variationnels
- MCMC méthodes

Approximate Inference Techniques

- Variational Methods
- Markov Chain Monte Carlo (MCMC) Methods
- Min-cut, Max Flow / Graph Cuts
- Exponential Family Models

Échantillonnage de Gibbs

Fonction GIBBS(X = {Z, E=e} , *bn*)
  retourne un échantillonne de P(Z|E=e)

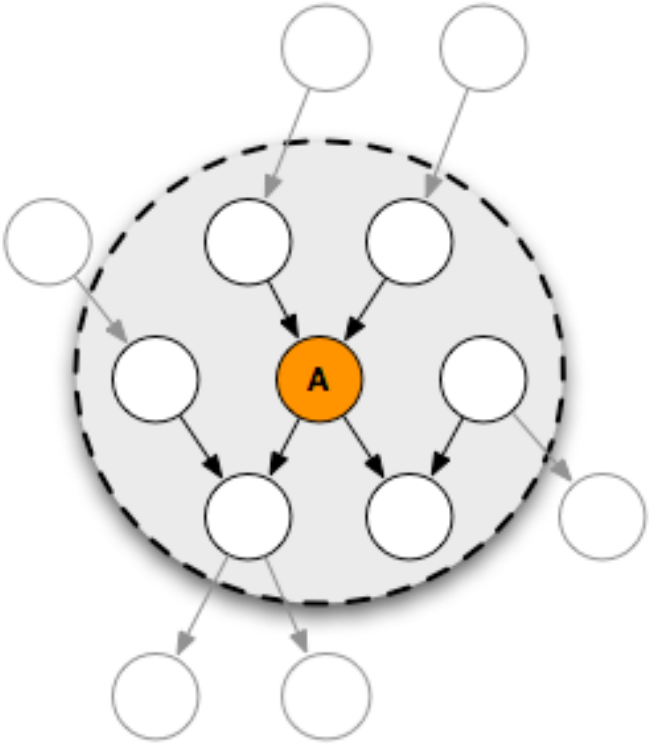
Variables locales:
  Z, les variables non observées dans bn
  x, l'état courant du réseau

Initialiser x = {z, e} avec des valeurs aléatoire pour Z=z

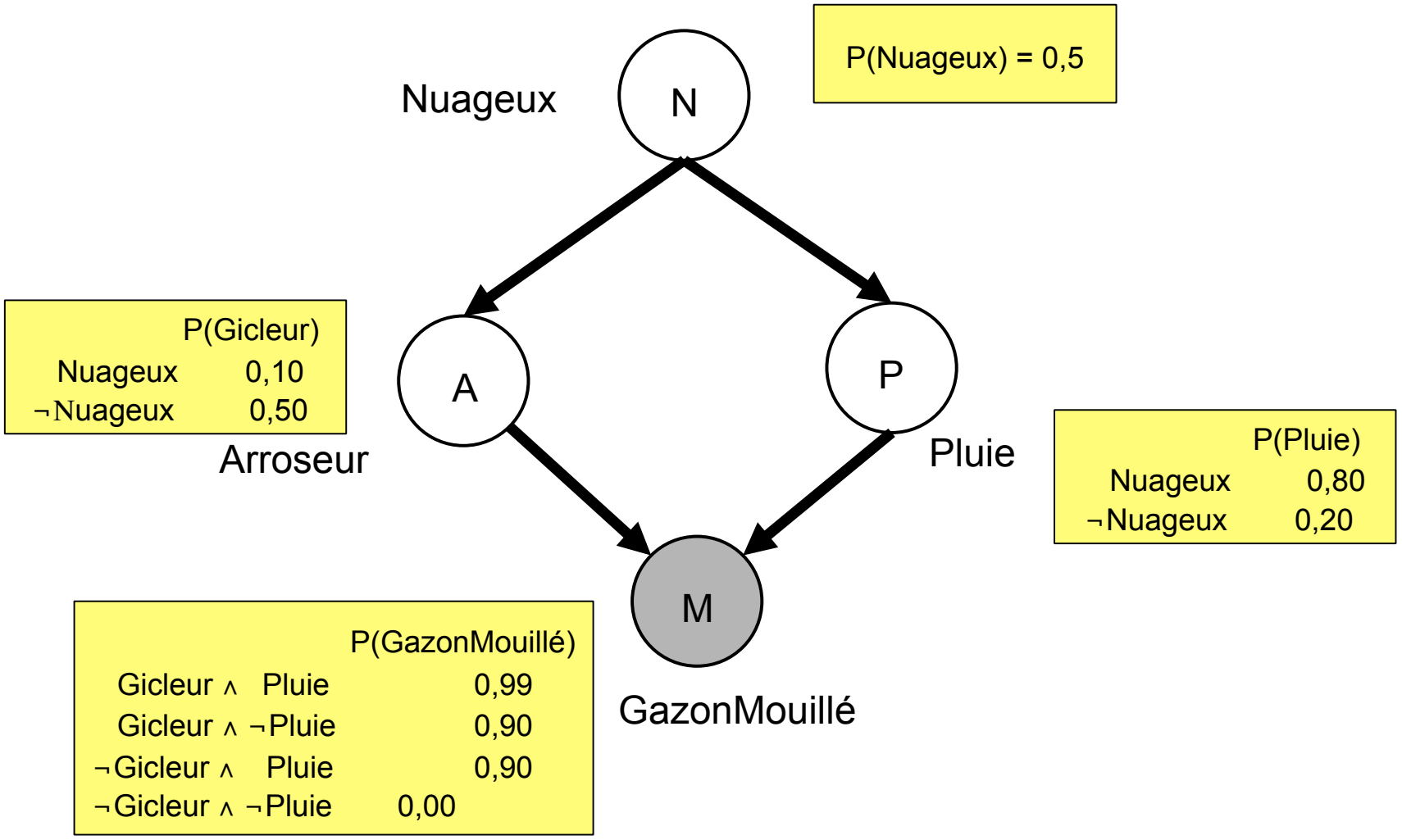
Pour j=1 à N faire
  pour chaque Z_i dans Z faire
    calculer la valeur de Zi dans x
    par échantillonnage de $P(Z_i|mb(Z_i))$

Rappel: couverture de Markov

- Etant donné un nœud, la couverture de Markov consiste de ses parents, ses fils et les parents de ses fils.



Echantilloner P(A,P,N|M=vraie)



Inférence – algorithmes approximatifs

- Échantillonnage par rejet:
  - Soit X la variable de la requête et **e** un ensemble de variables dont la valeur est spécifiée
  - on génère des échantillons
  - pour chaque échantillon compatible avec l'évidence **e**, on incrémente le compteur pour la valeur de X dans cet échantillon
  - à la fin, on normalise les quantités obtenues

Échantillonnage

x ← événement de *n* éléments (ordonnés selon l'ordre topologique)

pour *i* = 1 à *n* **faire**

$x_i \leftarrow$ une valeur aléatoire selon $P(X_i \mid Parents(X_i))$

retourner x

Échantillonnage par rejet

fonction ECHANT-REJET(X,e,réseau,*N*)

pour *i* = 1 à *N* **faire**

x ← ECHANTILLON(réseau)

si x est consistent avec e alors

N[*x*] ← N[*x*]+1, où *x* est la valeur de X dans x

retourner NORMALISER(N[X])

Inférence – algorithmes approximatifs

- Problème
  - Rejette beaucoup d'échantillons
  - Si la probabilité de **e** est faible, on peut se retrouver avec un ensemble d'échantillons peu représentatifs

Inférence – algorithmes approximatifs

- Pondération par vraisemblance
  - Tous les échantillons sont considérés
  - Pour construire un échantillon, on ne génère que les valeurs des variables qui ne sont pas dans **e, en parcourant les variables dans l'ordre topologique**
  - Pour chaque variable déjà dans ***e***, on calcule un poids qui dépend de la probabilité de la valeur, et tous ces poids sont multipliés
  - Pour chaque échantillon, on incrémente le compteur pour la valeur de X dans cet échantillon en ajoutant la valeur du poids calculé pour cet échantillon

Inférence – algorithmes approximatifs

- Discussion
  - Plus efficace, puisqu'il utilise tous les échantillons
  - Les performances se dégradent s'il y a beaucoup de variables dans **e**

Autres approches

- *Raisonnement par défaut:* une conclusion est acceptée jusqu'à preuve du contraire
- *Théorie de Dempster-Shafer:* plutôt que de calculer la probabilité d'une proposition, on calcule la probabilité que l'évidence supporte cette proposition
- *Logique floue:* on utilise plutôt un degré d'appartenance à un ensemble

Sparse Message Passing

Chris Pal

Some Deeper Analysis

- Traditional energy function (compact notation)

$$F(\boldsymbol{x}, \boldsymbol{y}) = \sum_i U(x_i, \boldsymbol{y}) + \sum_{i \sim j} V(x_i, x_j, \boldsymbol{y})$$

- Construct a CRF

$$P(\boldsymbol{X} \mid \boldsymbol{y}) = \frac{1}{Z(\boldsymbol{y})} \exp(-F(\boldsymbol{X}, \boldsymbol{y}))$$

$$Z(\boldsymbol{y}) = \sum_{\boldsymbol{x}} \exp(-F(\boldsymbol{x}, \boldsymbol{y}))$$

A Variational Approximation

$$\begin{aligned}
\mathrm{KL}\left(Q\left(\boldsymbol{X}\right) \parallel P\left(\boldsymbol{X} \mid \boldsymbol{y}\right)\right) &= \sum_{\boldsymbol{x}} Q\left(\boldsymbol{x}\right) \log \frac{Q\left(\boldsymbol{x}\right)}{P\left(\boldsymbol{x} \mid \boldsymbol{y}\right)} \\
&= \sum_{\boldsymbol{x}} Q\left(\boldsymbol{x}\right) \log \frac{Q\left(\boldsymbol{x}\right) Z\left(\boldsymbol{y}\right)}{\exp\left(-F\left(\boldsymbol{x}, \boldsymbol{y}\right)\right)} \\
&= \left\langle F\left(\boldsymbol{x}, \boldsymbol{y}\right)\right\rangle_{Q(\boldsymbol{X})} - H\left(Q\left(\boldsymbol{X}\right)\right) + \log Z\left(\boldsymbol{y}\right).
\end{aligned}$$

- We use a simpler distribution, $Q\left(\boldsymbol{X}\right) = \prod_i Q\left(X_i\right)$
- Define our “variational free energy” as

$$\mathcal{L}\left(Q\left(\boldsymbol{X}\right)\right) = \left\langle F\left(\boldsymbol{x}, \boldsymbol{y}\right)\right\rangle_{Q(\boldsymbol{X})} - H\left(Q\left(\boldsymbol{X}\right)\right)$$

Minimizing the KL Divergence

- Which we can state as

$$\mathrm{KL}\left(Q\left(\boldsymbol{X}\right) \parallel P\left(\boldsymbol{X} \mid \boldsymbol{y}\right)\right) = \mathcal{L}\left(Q\left(\boldsymbol{X}\right)\right) + \log Z\left(\boldsymbol{y}\right)$$

- Since the KL Divergence ≥ 0, and

$$\mathcal{L}\left(Q\left(\boldsymbol{X}\right)\right) \geq -\log Z\left(\boldsymbol{y}\right)$$

- Since $Z(\boldsymbol{y})$ is constant for an observation $\boldsymbol{y}$
- Thus, minimizing the variational free energy will also minimize the KL Divergence

Sparse Mean Field

- Leading to classical mean field updates

$$Q^*(x_j) = \frac{1}{Z_j} \exp\left(-\left\langle F(\boldsymbol{x}, \boldsymbol{y})\right\rangle_{\prod_{i:i\neq j} Q(X_i)}\right)$$

- However, with a ***sparse*** update $Q'(X_j)$

$$\begin{aligned}\mathrm{KL}\left(Q(\boldsymbol{X}) \parallel P(\boldsymbol{X} \mid \boldsymbol{y})\right) = \mathrm{KL}\left(Q'(X_j) \parallel Q^*(X_j)\right) \\ + \log Z_j + \log Z(\boldsymbol{y}) - \sum_{i:i\neq j} H\left(Q(X_i)\right)\end{aligned}$$

- Where the last 3 terms are constant w.r.t. $Q'(X_j)$
- and we can therefore …

Minimizing the Global Divergence

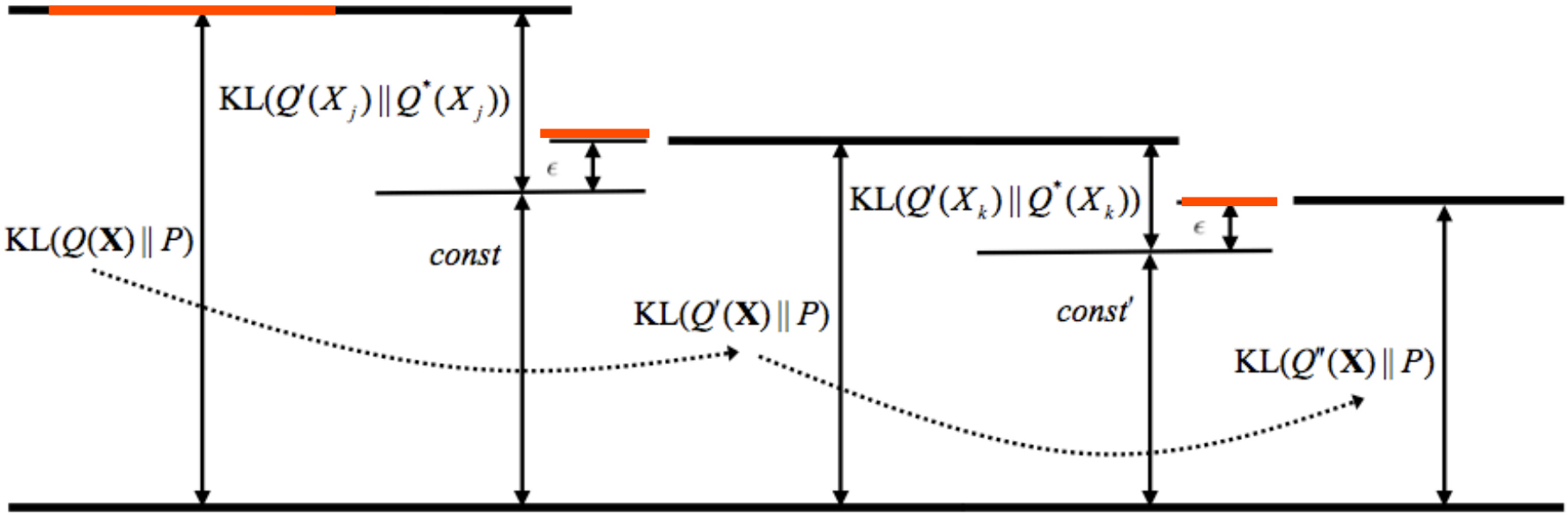


Fig. 1. Minimizing the global KL divergence via two different sparse local updates. The *global* divergence $\mathrm{KL}(Q(\mathbf{X}) \parallel P)$ can be decomposed into a *local* update plus a constant: $\mathrm{KL}(Q'(X_j) \parallel Q^*(X_j)) + const$. Consequently, at each step of sparse variational message passing we may minimize different local divergences to within some ϵ and when updating different local Qs, we minimize the global KL divergence.

Note: Szeliski *et al.* Comparative Study of Energy Minimization Methods for MRFs

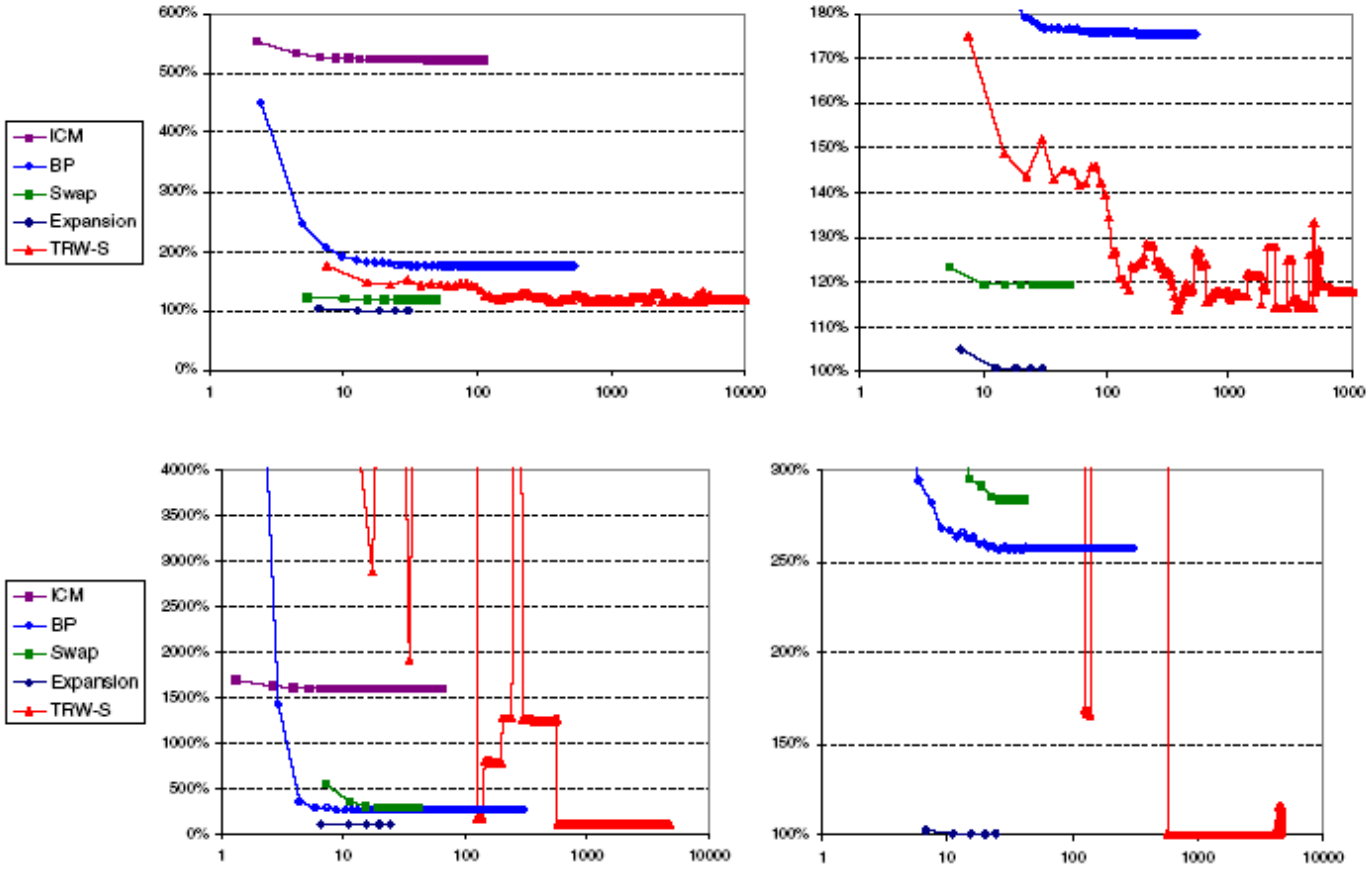


Fig. 2. Results on the Photomontage benchmarks, "Panorama" is at top and "Family" is below. Each plot shows energies vs. run time in seconds, using a log scale for time. The plots on the right are zoomed versions of the plots on the left. ICM is omitted in the right plots, due to poor performance. The associated color images can be found on the project web page.

Fast Free Energy Minimization

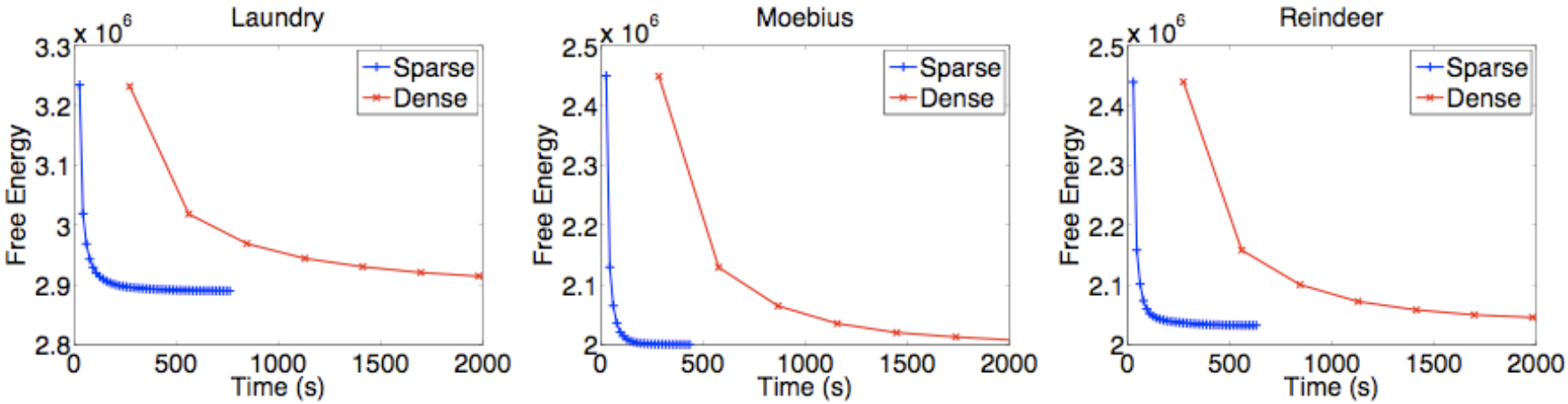


Fig. 3. Comparison of CPU time for free energy minimization with sparse and dense mean field updates using parameters Θ_v learned in the canonical model with three images (Art, Books, Dolls).

Fast Energy Minimization

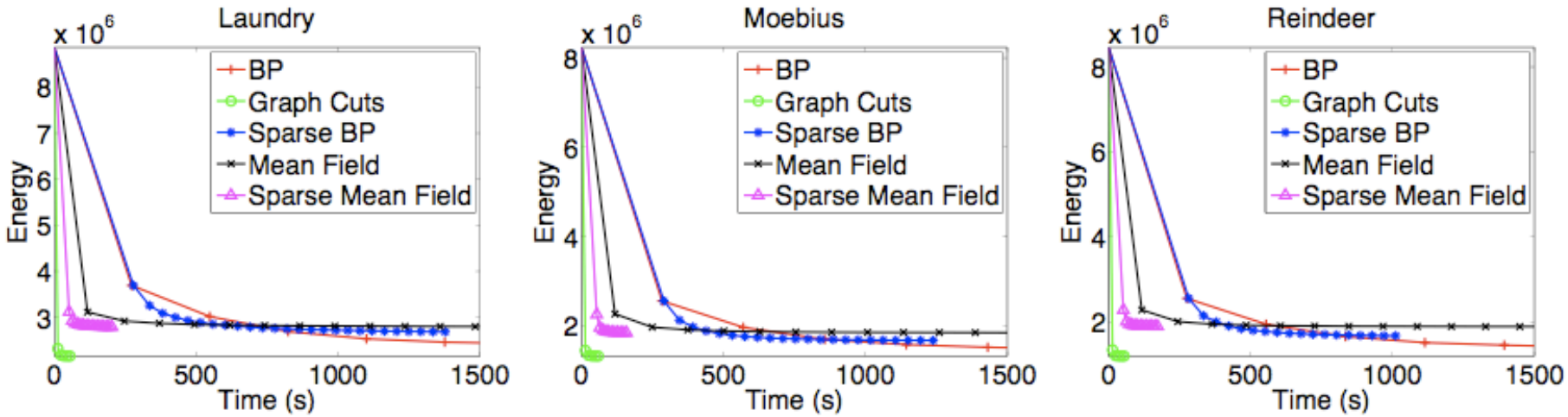


Fig. 4. CPU time versus energy for graph cuts, sum-product belief propagation, and mean field using parameters Θ_v learned with three images (Art, Books, Dolls). *Maximum posterior marginal* (MPM) prediction is used with the approximate marginal at each iteration.

Improved Learning Efficiency

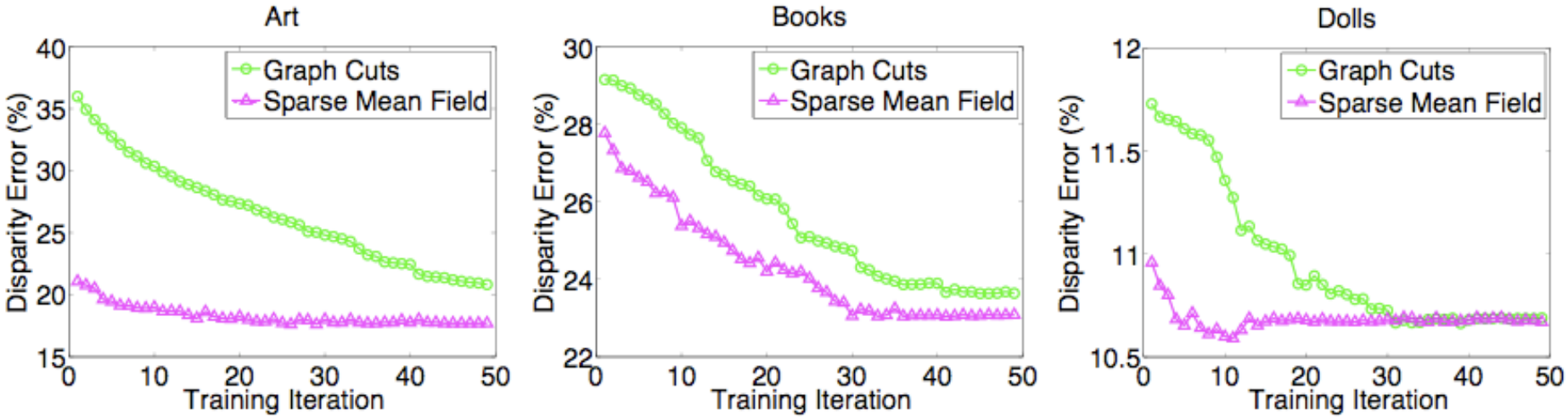


Fig. 6. Disparity error (each image held out in turn) using both graph cuts and mean field for learing the canonical CRF stereo model. The error before learning is omitted from the plots to better highlight performance differences.

The First Step Is Much Better

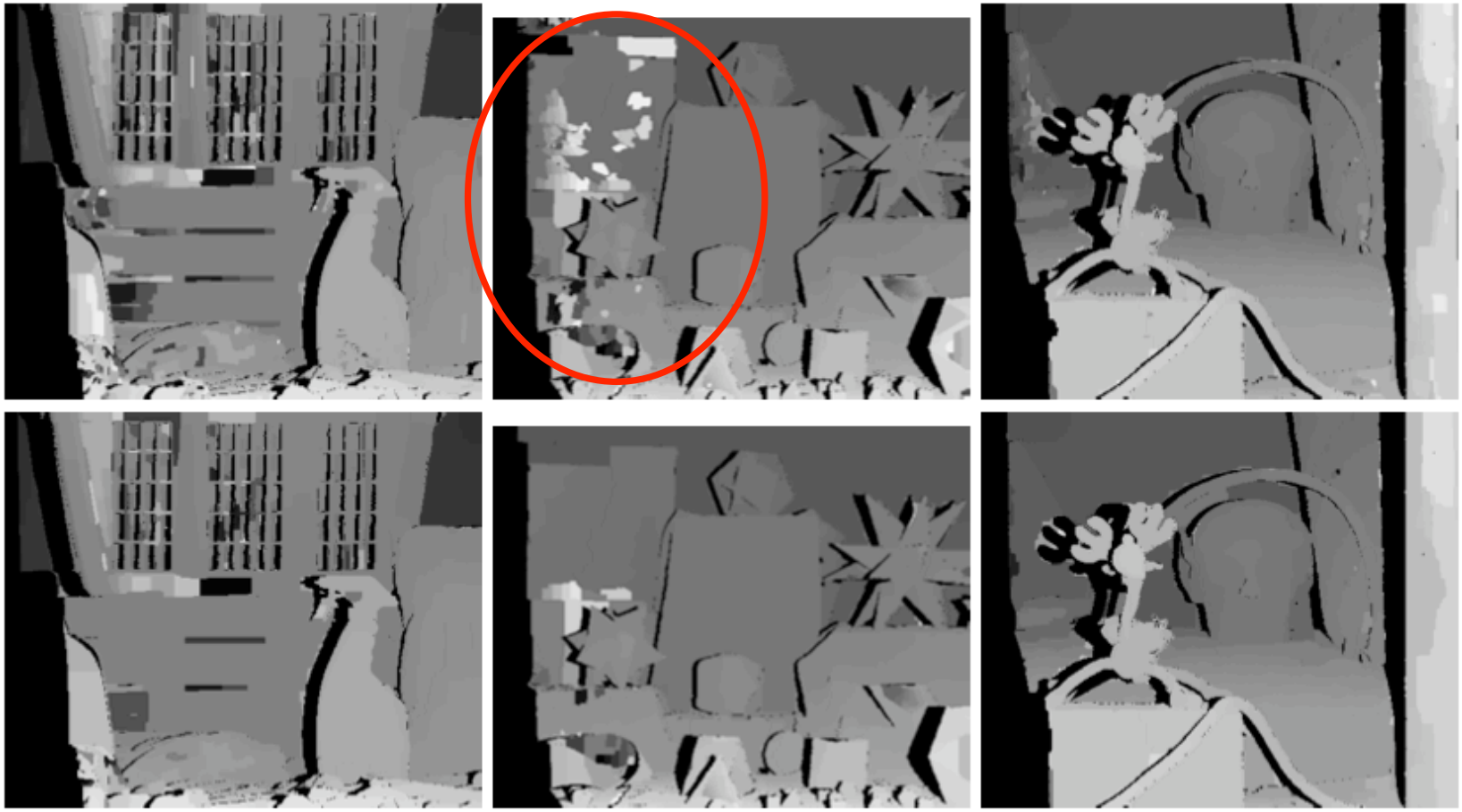


Fig. 5. Test images comparing prediction (using graph cuts) after one round of learning the canonical model with graph cuts (top) or sparse mean field (bottom). Occluded areas are black. Images (l-r): Laundry, Moebius, Reindeer.

Results 1:
Simple Canonical Model
Leave one out, training & testing

Metric	Method	Art	Books	Dolls	Laundry	Moebius	Reindeer	Average
Disparity Error	Graph Cuts	20.83	23.64	10.69	30.04	15.80	14.13	19.17
	Sparse Mean Field	17.70	23.08	10.67	29.16	15.43	13.37	18.22

Disparity error: % of incorrectly predicted pixels

Richer Models

- Gradient modulated occlusion boundaries

$$x_i \in \{0, \ldots, N-1\} \vee \text{“occluded”}$$

$$V(x_i, x_j, \boldsymbol{y}) = \begin{cases} 0 & \text{if } x_i = x_j \text{ and } x_i \neq \text{“occluded”} \\ \theta_k & \text{if } x_i \neq x_j,\ g_{ij} \in B_k \text{ and both } x_i, x_j \neq \text{“occluded”} \\ \theta_{o,o} & \text{if } x_i = x_j \text{ and } x_i = \text{“occluded”} \\ \theta_{o,k} & \text{if } x_i \neq x_j,\ g_{ij} \in B_k \text{ and } x_i \text{ or } x_j = \text{“occluded”}. \end{cases}$$

$$U(x_i, \boldsymbol{y}) = \begin{cases} c_i(x_i) & \text{if } x_i \neq \text{“occluded”} \\ \theta_o & \text{if } x_i = \text{“occluded”}, \end{cases}$$

Results 2:
Gradient Modulated Occlusion Model
Leave one out, training & testing

Metric	Method	Art	Books	Dolls	Laundry	Moebius	Reindeer	Average
Disparity Error	Graph Cuts	21.82	24.10	11.94	27.54	11.08	16.74	19.30
	Sparse Mean Field	21.05	23.14	11.62	27.37	11.45	16.44	18.93
Occlusion Error	Graph Cuts	34.50	28.27	32.99	36.89	40.65	50.83	37.36
	Sparse Mean Field	31.19	27.84	31.51	35.37	38.68	48.39	35.50

Disparity error: % of incorrectly predicted pixels

Occlusion error: occlusion prediction error %

Results 3:
Gradient Modulated Occlusion Model
Train and test on all

Metric	Method	Art	Books	Dolls	Laundry	Moebius	Reindeer	Average
Disparity Error	Graph Cuts	10.61	19.2	5.98	20.95	7.15	5.53	12.78
	Sparse Mean Field	8.29	13.41	4.72	19.22	5.11	4.76	10.15
Occlusion Error	Graph Cuts	16.20	10.40	24.88	29.77	27.88	32.97	21.83
	Sparse Mean Field	10.47	8.10	19.43	23.04	21.10	27.31	16.43

Disparity error: % of incorrectly predicted pixels

Occlusion error: occlusion prediction error %

CRFs, Stereo Vision and Sparse Message Passing

- Shown how we can construct CRFs for stereo
- Possible to use Graph Cuts for learning
- More accurate learning with mean field
- Faster learning with sparse mean field
- Principled: minimizes global KL divergence
- Middle ground - more complex approximations
- No non-negativity restrictions on parameters
- Potential for other graphical models

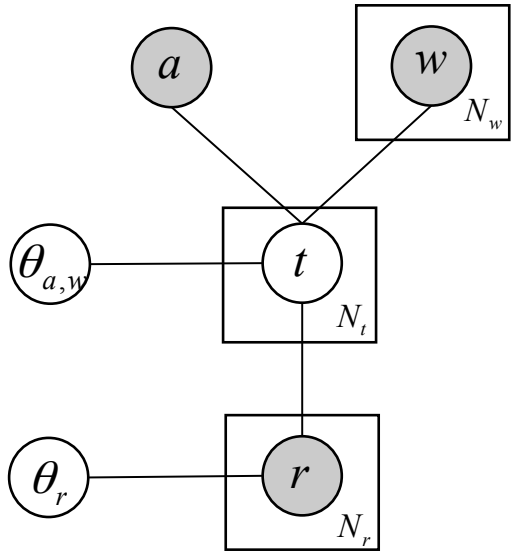
Transfer Learning with Conditional Boltzmann Machines

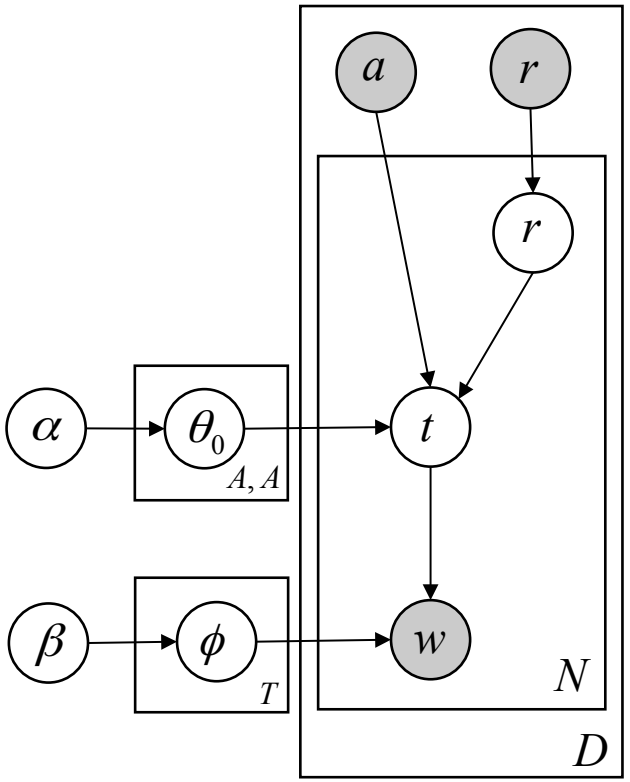
Chris Pal

Discriminative Author Recipient Topic (DART) Model

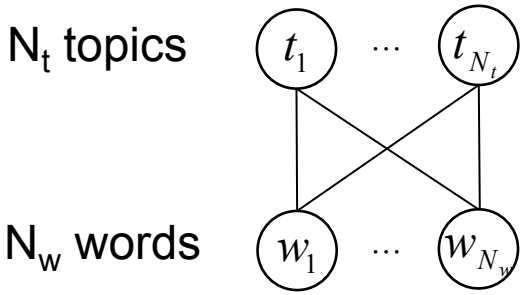
Directed, ART Model (Discrete)

Undirected, Continuous Topic DART Model

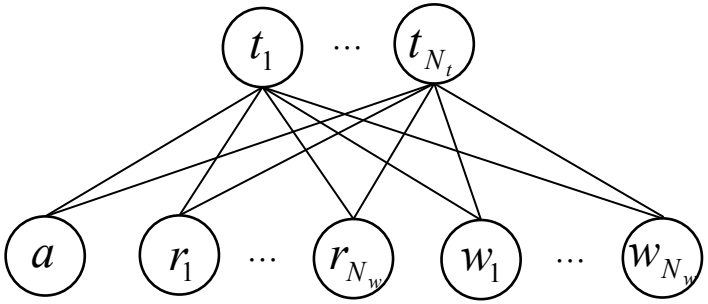




Undirected, Continuous Author Recipient Topic Models



1. A continuous topic model
2. Author recipient topic model
3. Plated version of same model



Author, N_r Recipients, Nw words

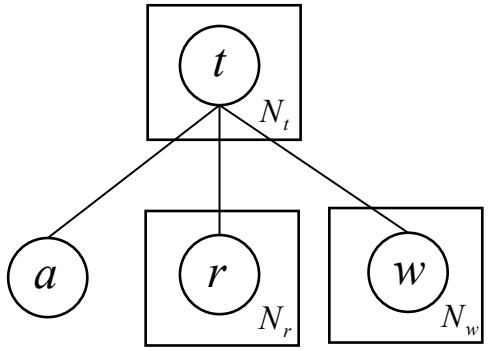


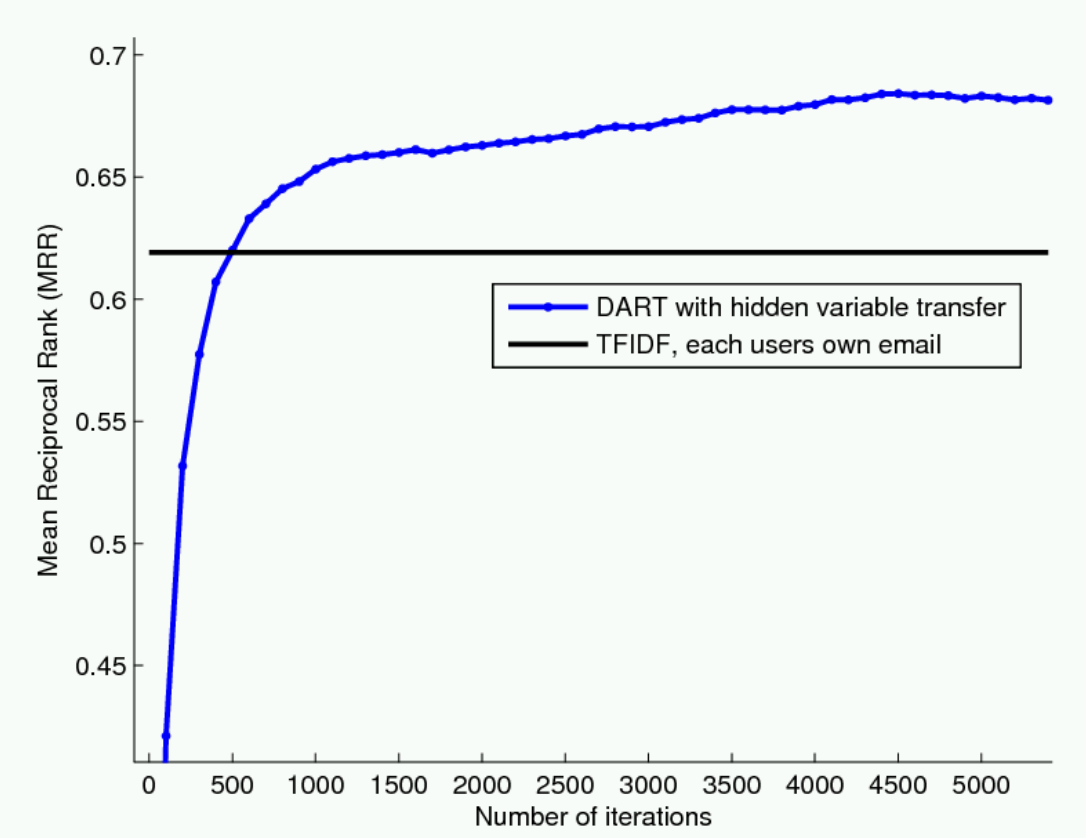
Plate Notation

Enron Email

- 150 employees
- 250,000 emails
- Avg. of 1400 sent emails [200 – 4800]

- Experiments with .9, .1 test-train split
- Use model to make prediction & cosine method
- Explore two types of transfer learning:
  1. Shared hidden variables
  2. Group and local models & coupled parameters

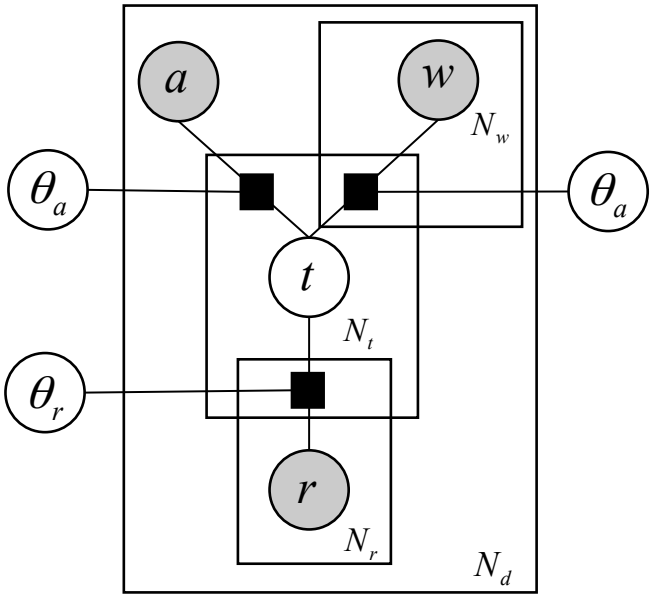
1. Transfer Using Shared Topics



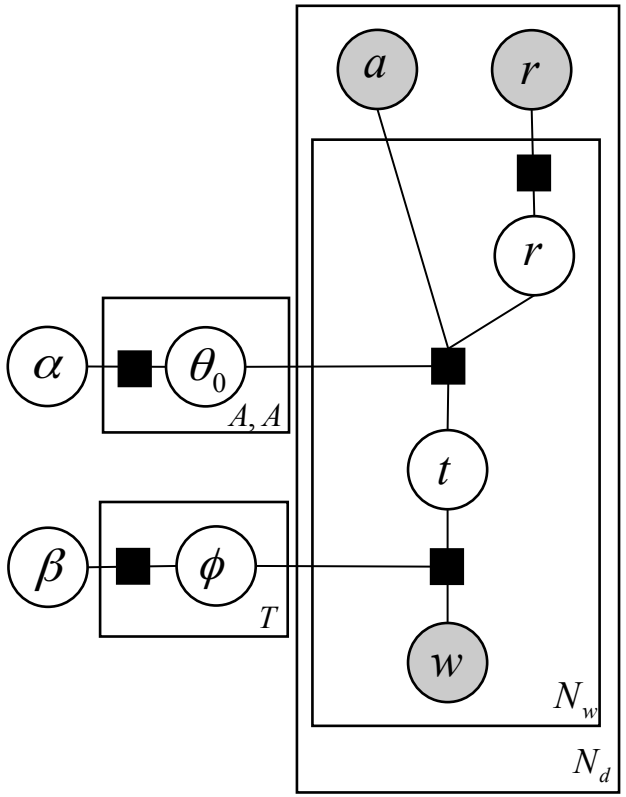
- Use model with shared latent space for predictions

Plated Factor Graphs for (D)ART Models

Undirected, Continuous Topic DART Model



Directed, Discrete Topic ART Model



■ - function

○ - unobserved variable

● - observed variable

DART Equations

- The underlying joint model

$$P(\mathbf{t_d}, \mathbf{w_d}, \mathbf{a_d}, c_d) \propto \exp(\sum_{i=1}^{T}(-\log(t_{di}) - \frac{1}{2}\log^2(t_{di}) + (\sum_{j=1}^{N_d} M^w_{iw_{dj}} + \sum_{k=1}^{S_d} M^a_{ia_{dk}} + M^c_{ic_d})\log(t_{di})))$$

- Optimize the marginal or marg. conditionals

$$\prod_{d=1}^{D} P(\mathbf{w_d}, \mathbf{a_d}, c_d)$$

$$\propto \exp(\frac{1}{2}\sum_{d=1}^{D}\sum_{i=1}^{T}(\sum_{j=1}^{V-1} M^w_{ij} m_{dj} + \sum_{k=1}^{S_d} M^a_{ia_{dk}} + M^c_{ic_d})^2)$$

DART Updates

- Gradient computation

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^T} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left(\mathbf{W}^T \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T - \frac{1}{N_s} \sum_{j=1}^{N_s} \mathbf{W}^T \tilde{\mathbf{x}}_{i,(j)} \tilde{\mathbf{x}}_{i,(j)}^T \right)$$

- Discriminative version

$$\begin{aligned} \frac{\partial \mathcal{L}_{MC}}{\partial \theta} = N \Bigg[(\alpha + \beta) \Bigg[\Bigg\langle \frac{\partial \mathbf{F}(\mathbf{x}_b, \mathbf{x}_d; \theta)}{\partial \theta} \Bigg\rangle_{\tilde{P}(\mathbf{x}_b, \mathbf{x}_d)} \\ - \alpha \Bigg\langle \Bigg\langle \frac{\partial \mathbf{F}(\mathbf{x}_b, \mathbf{x}_d; \theta)}{\partial \theta} \Bigg\rangle_{P(\mathbf{x}_d | \mathbf{x}_b; \theta)} \Bigg\rangle_{\tilde{P}(\mathbf{x}_b)} \Bigg] \\ - \beta \Bigg\langle \Bigg\langle \frac{\partial \mathbf{F}(\mathbf{x}_b, \mathbf{x}_d; \theta)}{\partial \theta} \Bigg\rangle_{P(\mathbf{x}_b | \mathbf{x}_d; \theta)} \Bigg\rangle_{\tilde{P}(\mathbf{x}_d)} \Bigg] \Bigg] \end{aligned}$$

Gibbs Sampler

- Recall - Log-normal & Discrete Conditionals

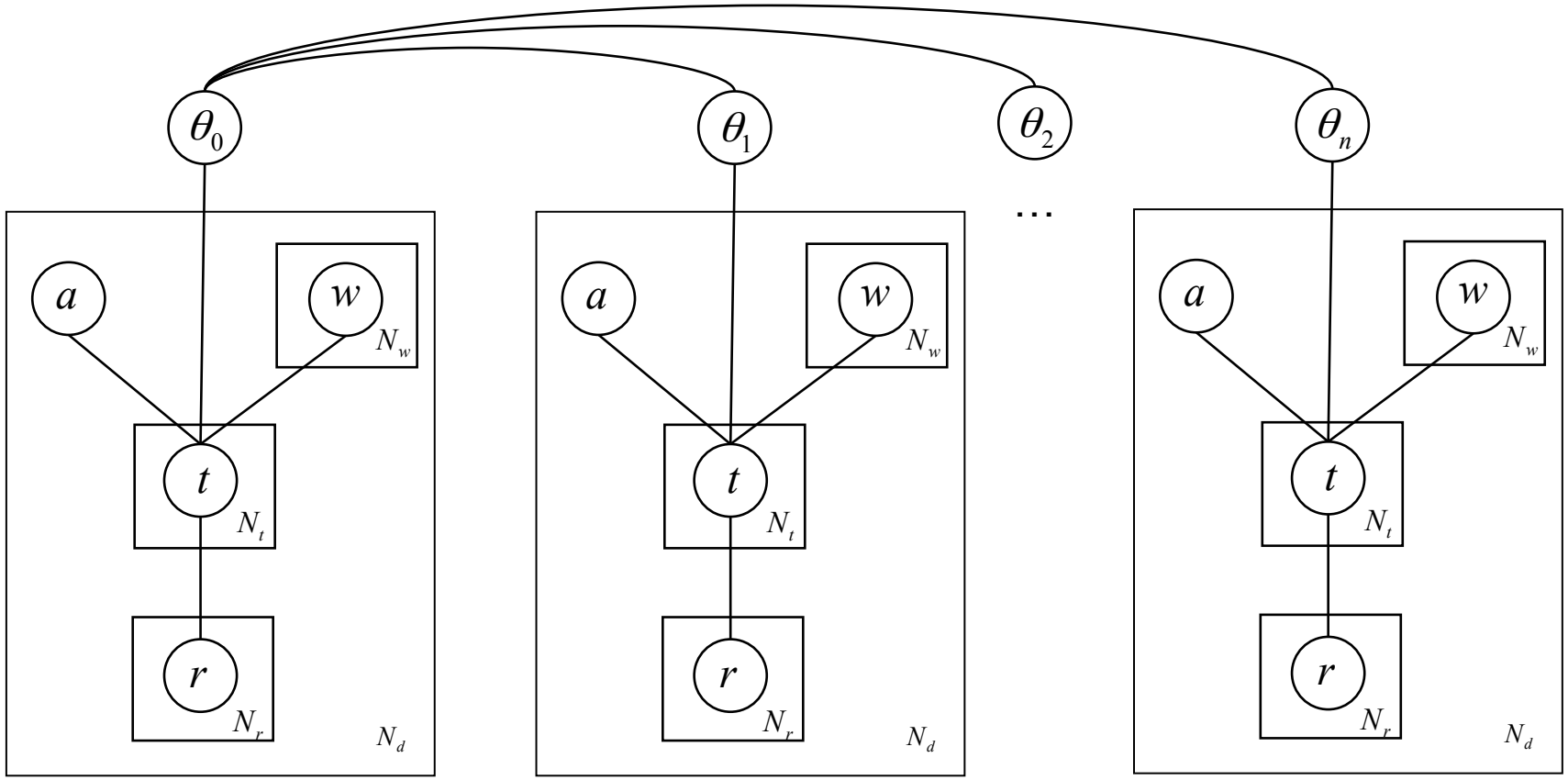
$$
\begin{aligned}
& p(t_{di}|\mathbf{w_d}, \mathbf{a_d}, c_d) \\
= \quad & \text{Log-normal}(\sum_{j=1}^{N_d} M^w_{iw_{dj}} + \sum_{k=1}^{S_d} M^a_{ia_{dk}} + M^c_{ic_d}, 1) \\
= \quad & \text{Log-normal}(\sum_{j=1}^{V-1} M_{ij} m_{dj} + \sum_{k=1}^{S_d} M^a_{ia_{dk}} + M^c_{ic_d}, 1)
\end{aligned}
$$

$$P(w_{dj}|\mathbf{t_d}) = \text{Discrete}(\sum_{i=1}^{T} \log(t_{di}) M^w_{i.})$$

$$P(a_{dk}|\mathbf{t_d}) = \text{Discrete}(\sum_{i=1}^{T} \log(t_{di}) M^a_{i.})$$

$$P(c_d|\mathbf{t_d}) = \text{Discrete}(\sum_{i=1}^{T} \log(t_{di}) M^c_{i.})$$

Transfer Learning with DARTs

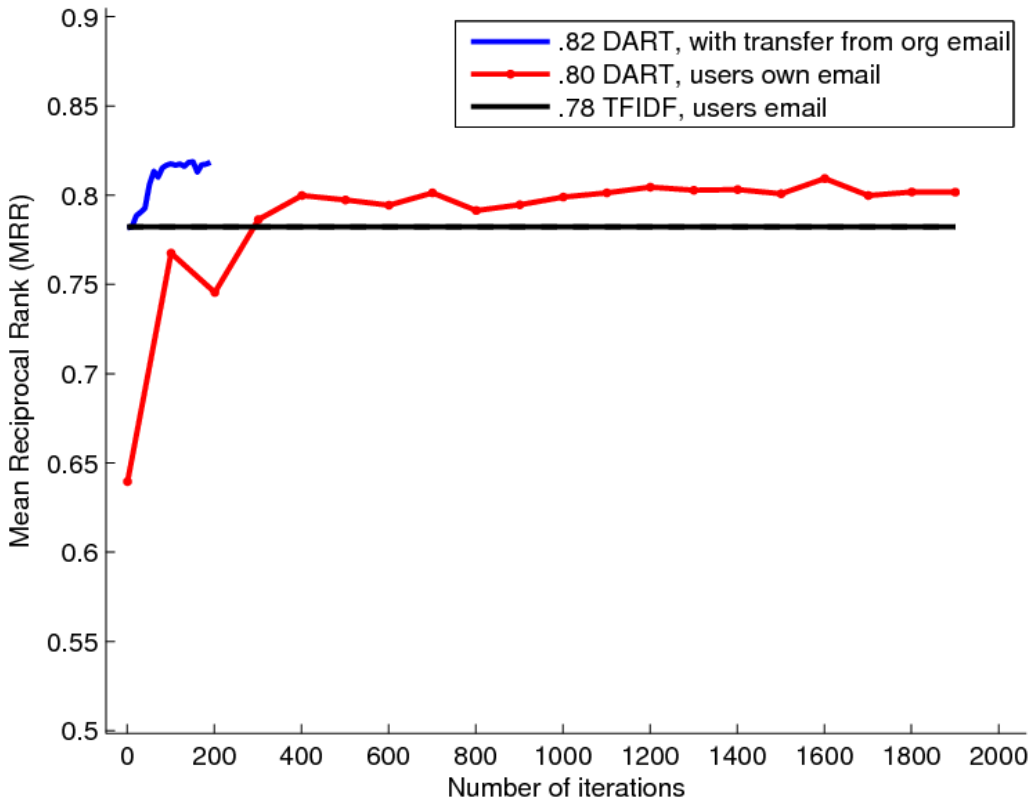


1. Train DART on orgs entire email

2. Adapt DART to a given users email

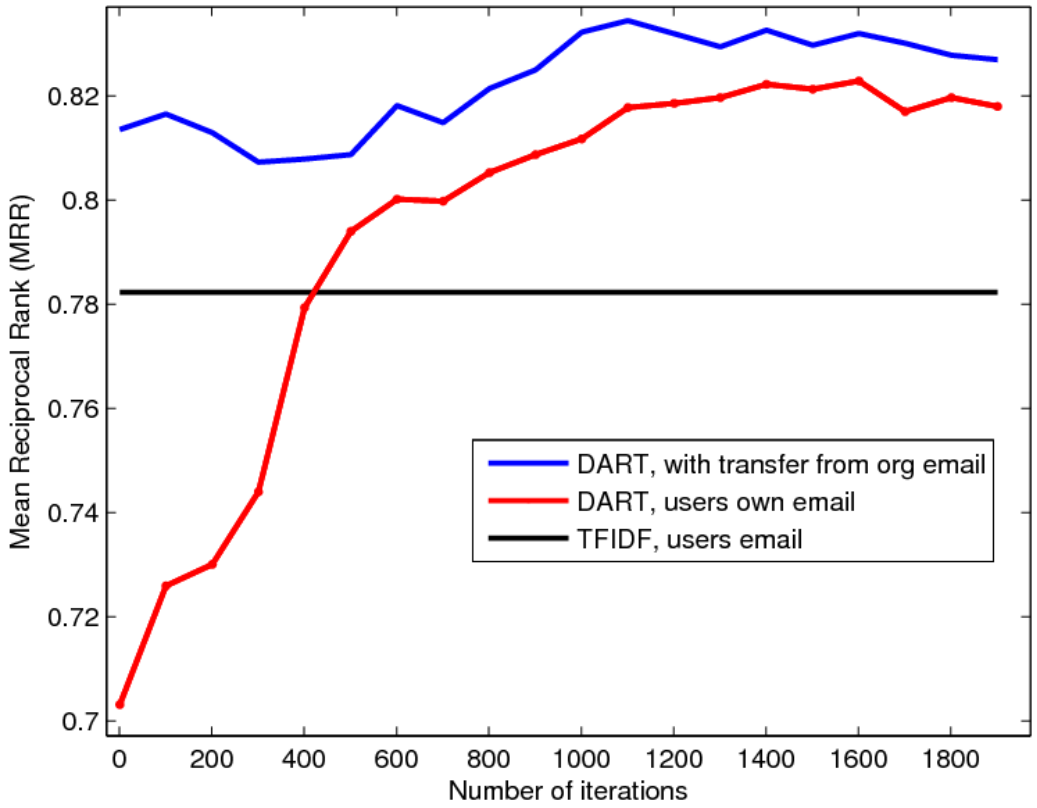
3. Major advantage for new users

2. Transfer Parameters & Adapt



- Topics with Transfer vs. No Transfer

Transfer Parameters & Adapt



- 200 Topic Models, Transfer vs. No Transfer

Summary, Conclusions Discussion

- New, rich topic models for text & attributes
- Discriminative methods - dramatic increase in task performance
- Two types of transfer learning
  - Each leverage social / org. networks

- Dramatic benefit for a new model/user
  Question: Can similar users be identified for more sophisticated transfer?
- Practical Issues: Information Sharing etc.

Résumé / Conclusions

- Présenter des nouvelles techniques d'optimisation qui contribuent à la création de nouvelles formes de médias interactifs
- Les modèles probabilistes et les champs aléatoires conditionnels en particulier sont des techniques de base qui s'appliquent dans une variété de domaines, dont :
  - la classification de documents,
  - l'extraction d'information de textes,
  - la segmentation d'images,
  - l'évaluation de la structure 3D d'une scène.
- Pour s'amuser – des vidéos