Reconnaissance de formes II

Réduction des attributs – Réseaux de neurones

Yves Goussard

GBM6103A

26 novembre 2014

Plan

Réduction du nombre d'attributs

Position du problème

- Vecteurs d'attributs $\boldsymbol{x}$ de taille N
- Si N est grand, lourdeur des calculs et possible détérioration des performances du classificateur

Objectifs

- Réduction de la taille des attributs $N \rightarrow M < N$
- Préservation, voire amélioration de la classification

Cadre

- I classes ω_i ; $1 \leq i \leq I$
- Pour chaque classe, $\boldsymbol{m}_i$ (moyenne) et $\boldsymbol{R}_i$ (matrice de covariance) connues
- Réduction : $\boldsymbol{y} = \boldsymbol{T}\boldsymbol{x}$; $\boldsymbol{y}$ de taille $M < N$; $\boldsymbol{T}$: transformation linéaire

Approche par minimum d'entropie (1)

Approche

Choisir $\boldsymbol{T}$ pour minimiser la dispersion dans chaque classe

Entropie

- f densité de probabilité ; $H(f) = -\mathrm{E}\left[\log f(x)\right] = -\int f(x) \log f(x) dx$
- Entropie $H(f)$: mesure la dispersion de f
- f uniforme de support de taille L :
$$H(f) = \log L$$
- f gaussienne multivariée $\mathcal{N}(\boldsymbol{m}, \boldsymbol{R})$:
$$H(f) = \frac{N}{2} + \frac{1}{2} \log |\boldsymbol{R}| + \frac{N}{2} \log 2\pi$$

Approche par minimum d'entropie (2)

Hypothèses

- $f(\boldsymbol{x}|\omega_i)$ gaussienne $\mathcal{N}(\boldsymbol{m}_i, \boldsymbol{R}_i)$
- Toutes les classes sont de covariance identique :

$$\forall i \in \{1, 2 \dots I\} : \boldsymbol{R}_i = \boldsymbol{R}$$

Résultat

- Dans chaque classe : $f(\boldsymbol{y}|\omega_i)$ gaussienne $\mathcal{N}(\boldsymbol{T}\boldsymbol{m}_i, \boldsymbol{T}\boldsymbol{R}\boldsymbol{T}^{\mathrm{t}})$
- Après transformation :

$$H_i = \frac{M}{2}(1 + \log 2\pi) + \frac{1}{2}\sum_{j=1}^{M} \log \lambda_j \quad ; \quad \lambda_j : \text{ val. propres de } \boldsymbol{T}\boldsymbol{R}\boldsymbol{T}^{\mathrm{t}}$$

- Sous condition de normalisation, H_i minimisée pour $\boldsymbol{T}$ composée des M vecteurs propres de $\boldsymbol{R}$ associés aux valeurs propres les plus petites

Discriminant linéaire de Fisher (1)

Approche

Choisir $\boldsymbol{T}$ pour maximiser le rapport entre dispersion inter-classes et dispersion intra-classe

Hypothèses

- Pour chaque classe, $\{\boldsymbol{m}_i, \boldsymbol{R}_i\}$ connues
- $\boldsymbol{R} = \sum_{i=1}^{I} \boldsymbol{R}_i \qquad \boldsymbol{m} = \frac{1}{I} \sum_{i=1}^{I} \boldsymbol{m}_i$

Démarche

- Dispersion intra-classe : $\boldsymbol{R}$
- Dispersion inter-classes : $\boldsymbol{B} = \sum_{i=1}^{I} (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^{\mathrm{t}}$

Discriminant linéaire de Fisher (2)

Démarche

- Rapport des dispersions avant transformation : $|\boldsymbol{B}|/|\boldsymbol{R}|$
- Rapport des dispersions après transformation : $|\boldsymbol{T}\boldsymbol{B}\boldsymbol{T}^{\mathrm{t}}|/|\boldsymbol{T}\boldsymbol{R}\boldsymbol{T}^{\mathrm{t}}|$
- Quantité maximisée pour $\boldsymbol{T}$ composée des M vecteurs propres de $\boldsymbol{R}^{-1}\boldsymbol{B}$ associés aux valeurs propres les plus grandes

Remarque

On doit choisir $M < I$ pour que $\boldsymbol{B}$ soit inversible

Décomposition en composantes principales

Approche

- Meilleure approximation de $\boldsymbol{x}$ par $\boldsymbol{y}$ au sens de l'erreur quadratique moyenne
- Sélection de $\boldsymbol{y}$ de manière à maximiser la dispersion *globale*

Démarche

- Dispersion avant transformation : $|\boldsymbol{R}|$
- Dispersion après transformation : $|\boldsymbol{T}\boldsymbol{R}\boldsymbol{T}^{\mathrm{t}}|$
- Sous condition de normalisation, quantité maximisée pour $\boldsymbol{T}$ composée des M vecteurs propres de $\boldsymbol{R}$ associés aux valeurs propres les plus grandes

Remarques

- On utilise $f(\boldsymbol{x})$ et non $f(\boldsymbol{x}|\omega_i)$!
- Pas d'effet garanti sur la séparation entre classes

Classification par réseaux de neurones

Objectif général

Mise en œuvre simple d'une méthode de classification par minimum de distance quadratique

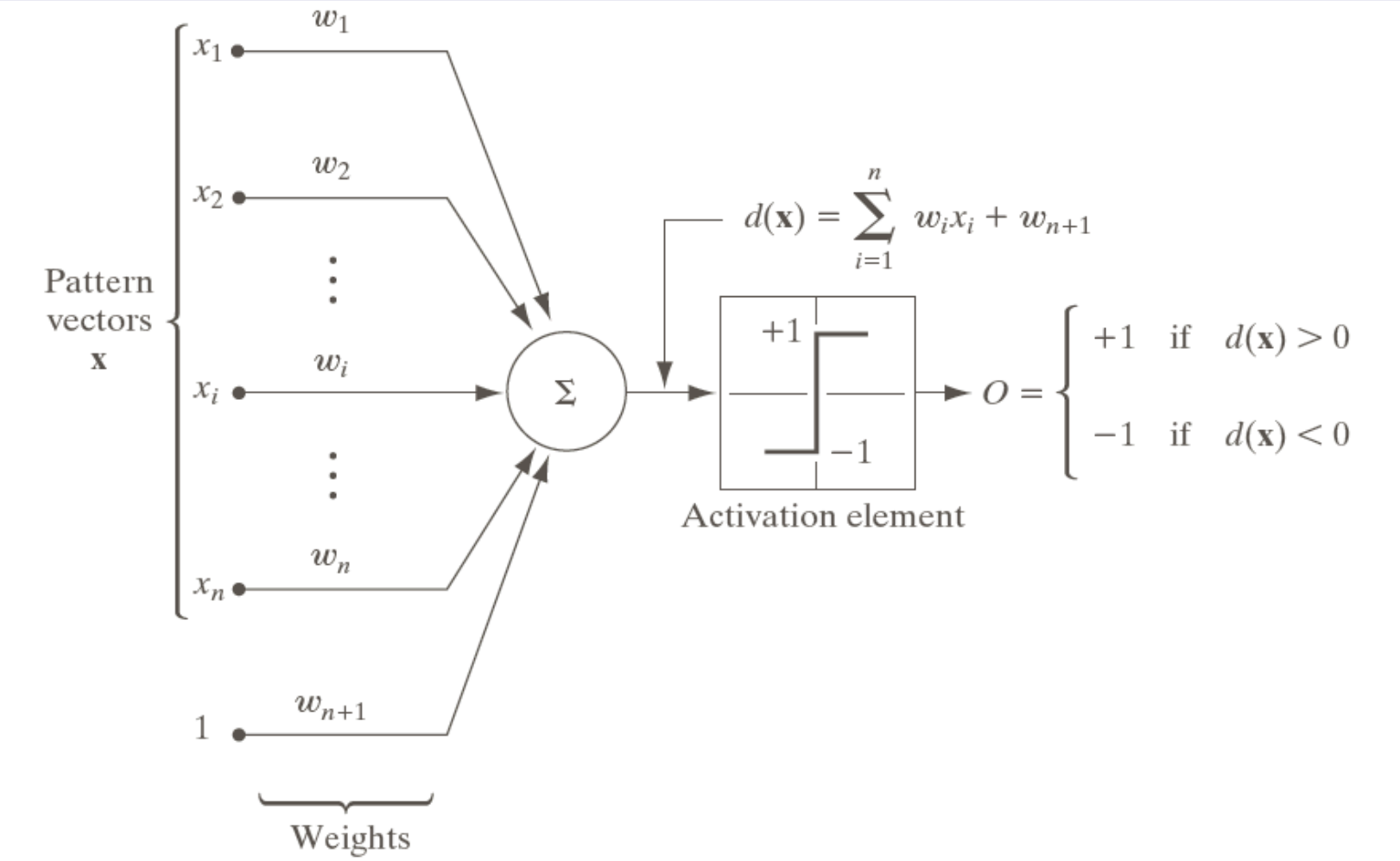
Perceptron

- Deux classes
- Distance quadratique : surface de décision = hyperplan

Structure du classificateur $\boldsymbol{w}^{\mathrm{t}}\boldsymbol{x} \underset{<}{\overset{>}{}} t$

Perceptron (1)

Structure



Perceptron (2)

Formalisation et entraînement

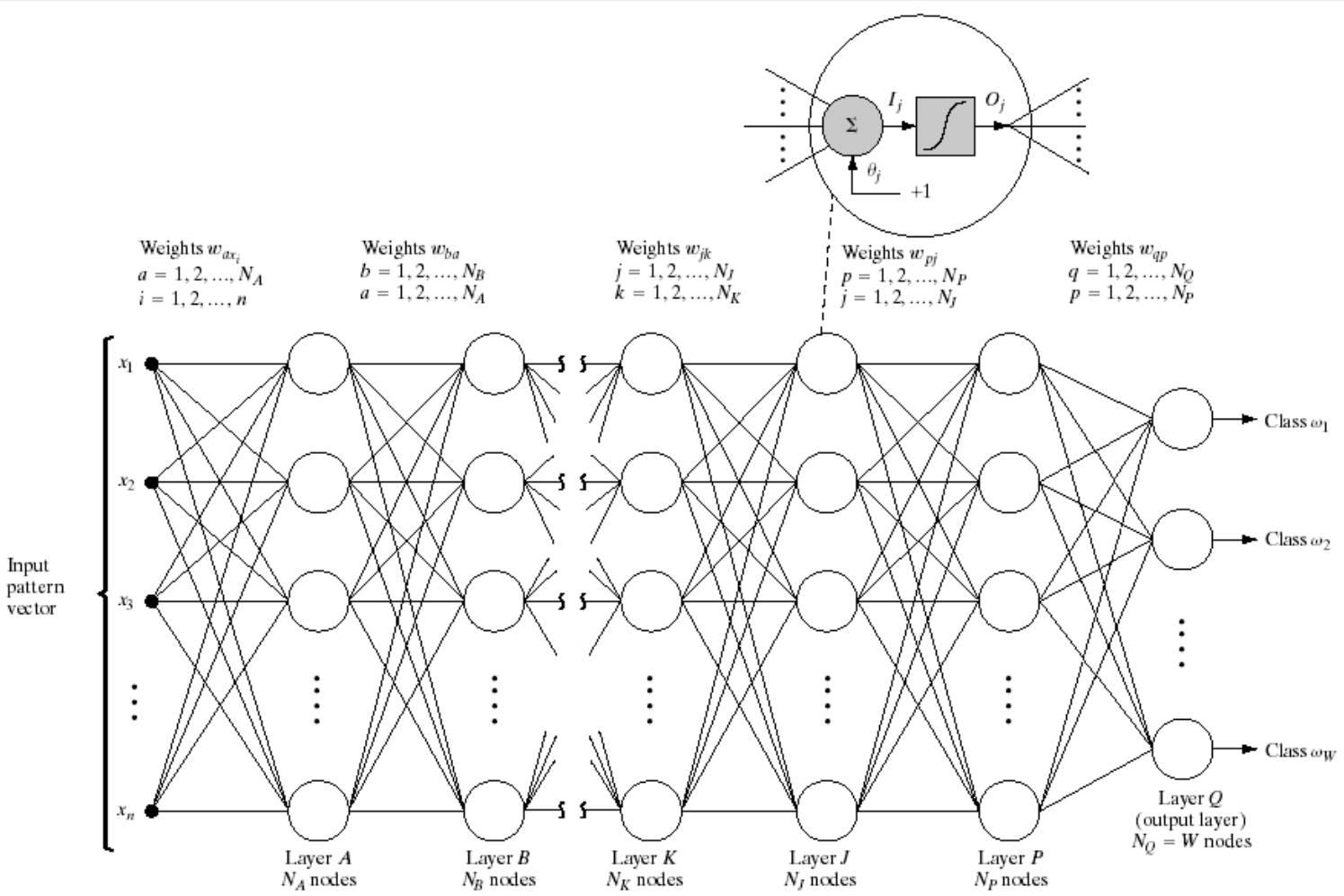
- Relation entrée-sortie : $\boldsymbol{y}^{\mathrm{t}}\boldsymbol{w} = r$
- Concaténation pour les éléments de l'ensemble d'apprentissage : $\boldsymbol{Y}^{\mathrm{t}}\boldsymbol{w} = \boldsymbol{r}$
- Estimation de $\boldsymbol{w}$ par moindres carrés : $\hat{\boldsymbol{w}} = (\boldsymbol{Y}^{\mathrm{t}}\boldsymbol{Y})^{-1}\boldsymbol{Y}^{\mathrm{t}}\boldsymbol{r}$
- Minimisation par algorithme du gradient :
$$\hat{\boldsymbol{w}}^{(i+1)} = \hat{\boldsymbol{w}}^{(i)} + \alpha \boldsymbol{y}_{i+1}(r_{i+1} - \boldsymbol{y}_{i+1}^{\mathrm{t}}\hat{\boldsymbol{w}}^{(i)})$$
- Divers choix possibles pour α, selon la difficulté du problème

Caractéristiques de l'approche

- Limitée en tant que telle
- « Brique de base » pour les réseaux de neurones

Réseaux à couches multiples (1)

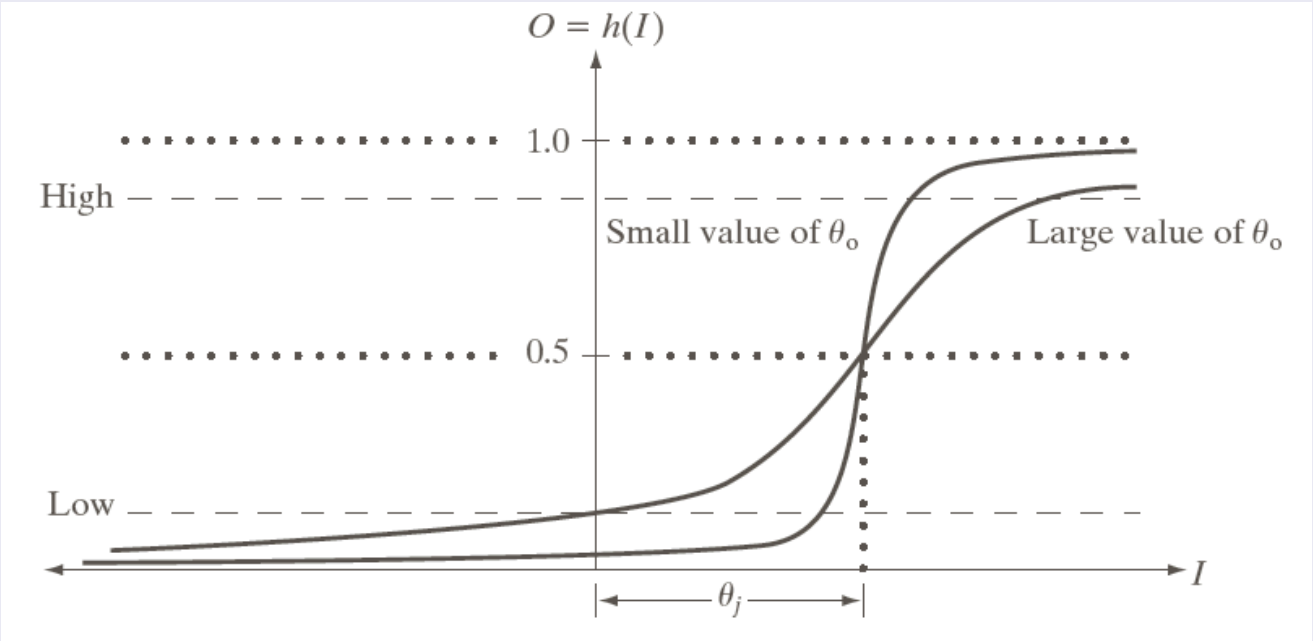
Structure



Réseaux à couches multiples (2)

Quantités à fixer

- Nombre de couches
- Nombre de cellules par couche
- Fonction seuil $O = h(I)$



Réseaux à couches multiples (3)

Entraînement : algorithme de rétropropagation

- Entraînement du perceptron :

$$\Delta w_n = -\alpha \frac{\partial E}{\partial w_n} \qquad E = \frac{1}{2}\left(r - \sum_{n=1}^{N+1} y_n w_n\right)^2$$

- Rétropropagation : extension par parcours du réseau de droite à gauche
- Dernière couche :

$$\Delta w_{qp} = \alpha \underbrace{(r_q - O_q) h'(I_q)}_{\delta_q} O_p \qquad E_q = \tfrac{1}{2}\sum (r_q - O_q)^2$$

- Avant-dernière couche : on ne connaît pas la sortie désirée r_p !

$$E_p = \frac{1}{2}\sum_p (r_p - O_p)^2$$

$$\text{mais} \quad \delta_p = \frac{\partial E_p}{\partial I_p} = h'(I_p)\sum_q \delta_q w_{qp} \quad \text{et} \quad \Delta w_{pj} = \alpha \delta_p O_j$$

Réseaux à couches multiples (4)

Avantages de l'approche

- Technique relativement simple
- Structure souple
- Fonctionnement de type boîte noire

Limitations de l'approche

- Structure parfois difficile à spécifier
- Parfois difficile à entraîner (mauvaise sensibilité des paramètres internes, non-linéarités, algorithme lent)
- Fonctions de décision : hyperplans par morceaux