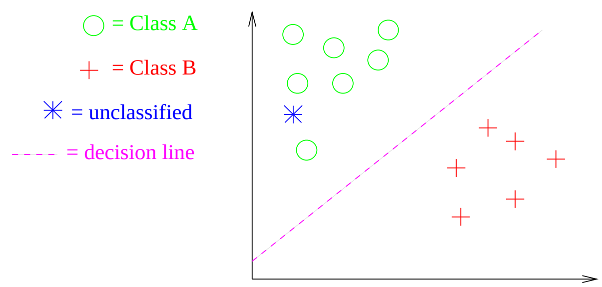
CSCE 970 Lecture 3:
Linear Classifiers

Stephen D. Scott

January 21, 2003

Introduction

- Sometimes probabilistic information unavailable or mathematically intractable
- Many alternatives to Bayesian classification, but optimality guarantee may be compromised!
- Linear classifiers use a decision hyperplane to perform classification
- Simple and efficient to train and use
- Optimality requires linear separability of classes



Linear Discriminant Functions

- Let $\mathbf{w} = [w_1, \ldots, w_\ell]^T$ be a weight vector and w_0 (a.k.a. θ) be a threshold
- Decision surface is a hyperplane:

$$\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$$

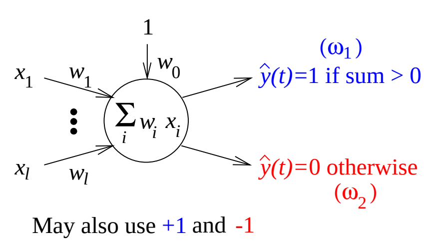
- E.g. predict ω_2 if $\sum_{i=1}^{\ell} w_i x_i > w_0$, otherwise predict ω_1
- Focus of this lecture: How to find w_i's
  - Perceptron algorithm
  - Winnow
  - Least squares methods (if classes not linearly separable)

The Perceptron Algorithm

- Assume linear separability, i.e. $\exists \mathbf{w}^*$ s.t.

$$\mathbf{w}^{*T} \cdot \mathbf{x} > 0 \quad \forall \mathbf{x} \in \omega_1$$
$$\mathbf{w}^{*T} \cdot \mathbf{x} \leq 0 \quad \forall \mathbf{x} \in \omega_2$$

  (w_0^* is included in $\mathbf{w}^*$)
- So $\exists$ deterministic function classifying vectors (contrary to Ch. 2 assumptions)

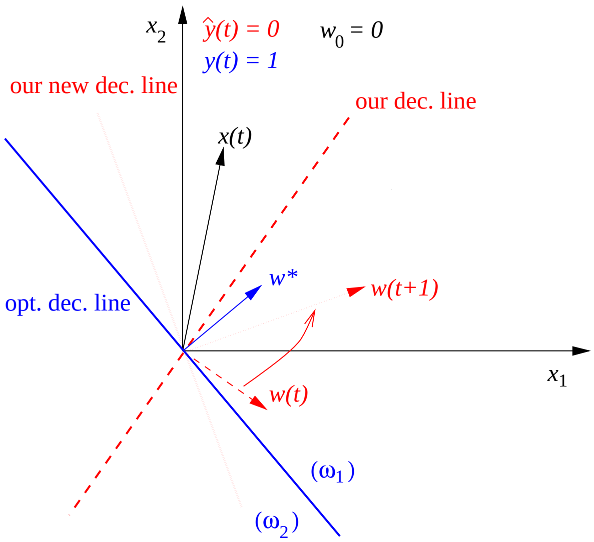


- Given actual label $y(t)$ for trial t, update weights:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \rho(y(t) - \hat{y}(t))\mathbf{x}(t)$$

  - $\rho > 0$ is learning rate
  - $(y(t) - \hat{y}(t))$ moves weights toward correct prediction for $\mathbf{x}$

The Perceptron Algorithm
Example



The Perceptron Algorithm
Intuition

- Compromise between correctiveness and conservativeness
  - Correctiveness: Tendency to improve on $\mathbf{x}(t)$ if prediction error made
  - Conservativeness: Tendency to keep $\mathbf{w}(t+1)$ close to $\mathbf{w}(t)$
- Use cost function that measures both:

$$U(\mathbf{w}) = \overbrace{\|\mathbf{w}(t+1) - \mathbf{w}(t)\|_2^2}^{conserv.} + \eta \overbrace{(y(t) - \mathbf{w}(t+1)\cdot\mathbf{x}(t))^2}^{corrective}$$
$$= \sum_{i=1}^{\ell} (w_i(t+1) - w_i(t))^2 + \eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t+1)\, x_i(t)\right)^2$$

The Perceptron Algorithm
Intuition
(cont'd)

- Take gradient w.r.t. $\mathbf{w}(t+1)$ and set to $\mathbf{0}$:

$$0 = 2\,(w_i(t+1) - w_i(t)) - 2\eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t+1)\, x_i(t)\right) x_i(t)$$

- Approximate with

$$0 = 2\,(w_i(t+1) - w_i(t)) - 2\eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t)\, x_i(t)\right) x_i(t),$$

  which yields

$$w_i(t+1) = w_i(t) + \eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t) x_i(t)\right) x_i(t)$$

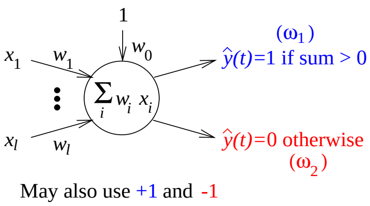
- Applying threshold to summation yields

$$w_i(t+1) = w_i(t) + \eta\,(y(t) - \hat{y}(t))\, x_i(t)$$

The Perceptron Algorithm
Miscellany

- If classes linearly separable, then by cycling through vectors, guaranteed to converge in finite number of steps
- For real-valued output, can replace threshold function on sum with
  - Identity function: $f(x) = x$
  - Sigmoid function: e.g. $f(x) = \frac{1}{1+\exp(-ax)}$
  - Hyperbolic tangent: e.g. $f(x) = c\tanh(ax)$

Winnow/Exponentiated Gradient



- Same as Perceptron, but update weights:

$$w_i(t+1) = w_i(t)\exp\left(-2\eta(\hat{y}(t) - y(t))\, x_i(t)\right)$$

- If $y(t), \hat{y}(t) \in \{0,1\}\, \forall t$, then set $\eta = (\ln\alpha)/2$ $(\alpha > 1)$ and get Winnow:

$$w_i(t+1) = \begin{cases} w_i(t)/\alpha^{x_i(t)} & \text{if } \hat{y}(t) = 1,\ y(t) = 0 \\ w_i(t)\alpha^{x_i(t)} & \text{if } \hat{y}(t) = 0,\ y(t) = 1 \\ w_i(t) & \text{if } \hat{y}(t) = y(t) \end{cases}$$

Winnow/Exponentiated Gradient
Intuition

- Measure distance in cost function with unnormalized relative entropy:

$$U(\mathbf{w}) = \overbrace{\sum_{i=1}^{\ell}\left(w_i(t) - w_i(t+1) + w_i(t+1)\ln\frac{w_i(t+1)}{w_i(t)}\right)}^{conserv.} + \eta\,\overbrace{(y - \mathbf{w}(t+1)\cdot\mathbf{x}(t))^2}^{corrective}$$

- Take gradient w.r.t. $\mathbf{w}(t+1)$ and set to $\mathbf{0}$:

$$0 = \ln\frac{w_i(t+1)}{w_i(t)} - 2\eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t+1)\, x_i(t)\right) x_i(t)$$

- Approximate with

$$0 = \ln\frac{w_i(t+1)}{w_i(t)} - 2\eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t) x_i(t)\right) x_i(t),$$

which yields

$$w_i(t+1) = w_i(t)\exp\left(-2\eta\left(\hat{y}(t) - y(t)\right) x_i(t)\right)$$

Winnow/Exponentiated Gradient
Negative Weights

- Winnow and EG update wts by multiplying by a pos const: impossible to change sign
  - Weight vectors restricted to one quadrant
- Solution: Maintain wt vectors $\mathbf{w}^+(t)$ and $\mathbf{w}^-(t)$
  - Predict $\hat{y}(t) = \left(\mathbf{w}^+(t) - \mathbf{w}^-(t)\right)\cdot\mathbf{x}(t)$
  - Update:

$$r_i^+(t) = \exp\left(-2\eta\left(\hat{y}(t) - y(t)\right) x_i(t)\, U\right)$$

$$r_i^-(t) = 1/r_i^+(t)$$

$$w_i^+(t+1) = U \cdot \frac{w_i^+(t)\, r_i^+(t)}{\sum_{j=1}^{\ell}\left(w_i^+(t)\, r_i^+(t) + w_i^-(t)\, r_i^-(t)\right)}$$

U and denominator normalize wts for proof of error bound

Kivinen & Warmuth, "Additive Versus Exponentiated Gradient Updates for Linear Prediction." *Information and Computation*, 132(1):1–64, Jan. 1997. [see web page]

Winnow/Exponentiated Gradient
Miscellany

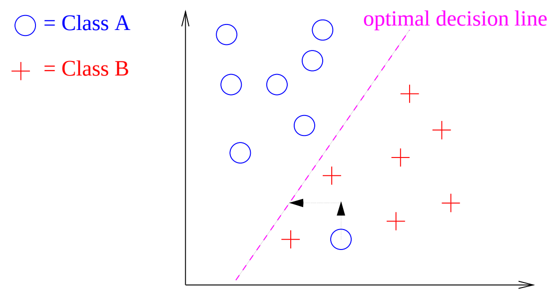
- Winnow and EG are muliplicative weight update schemes versus additive weight update schemes, e.g. Perceptron
- Winnow and EG work well when most attributes (features) are irrelevant, i.e. optimal weight vector $\mathbf{w}^*$ is sparse (many 0 entries)
- E.g. $x_i \in \{0,1\}$, $\mathbf{x}$'s are labelled by a monotone k-disjunction over ℓ attributes, $k \ll \ell$
  - Remaining $\ell - k$ are irrelevant
  - E.g. $x_5 \vee x_9 \vee x_{12}$, $\ell = 150$, $k = 3$
  - For disjunctions, number of on-line prediction mistakes is $O(k\log\ell)$ for Winnow and worst-case $\Omega(k\ell)$ for Perceptron
  - So in worst case, need exponentially fewer updates for training in Winnow than Perceptron
- Other bounds exist for real-valued inputs and outputs

Non-Linearly Separable Classes

- What if no hyperplane completely separates the classes?
- Add extra inputs that are nonlinear combinations of original inputs (Section 4.14)
  - E.g. attribs. x_1 and x_2, so try $\mathbf{x} = \left[x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^2x_2, x_1x_2^2, x_1^3, x_2^3\right]^T$
  - Perhaps classes linearly separable in new feature space
  - Useful, especially with Winnow/EG logarithmic bounds
  - Kernel functions/SVMs
- Pocket algorithm (p. 63) guarantees convergence to a best hyperplane
- Winnow's & EG's agnostic results
- Least squares methods (Sec. 3.4)
- Networks of classifiers (Ch. 4)

Non-Linearly Separable Classes
Winnow's Agnostic Results

- Winnow's total number of prediction mistakes loss (in on-line setting) provably not much worse than best linear classifier



- Loss bound related to performance of best classifier and total distance under $\|\cdot\|_1$ that feature vectors must be moved to make best classifier perfect [Littlestone, COLT '91]
- Similar bounds for EG [Kivinen & Warmuth]

Non-Linearly Separable Classes
Least Squares Methods

- Recall from Slide 7:

$$w_i(t+1) = w_i(t) + \eta\left(y(t) - \sum_{i=1}^{\ell} w_i(t)x_i(t)\right)x_i(t)$$
$$= w_i(t) + \eta\left(y(t) - \mathbf{w}(t)^T \cdot \mathbf{x}(t)\right)x_i(t)$$

- If we don't threshold dot product during training and allow η to vary each trial (i.e. substitute η_t), get* Eq. 3.38, p. 69:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t\, \mathbf{x}(t)\left(y(t) - \mathbf{w}(t)^T \cdot \mathbf{x}(t)\right)$$

- This is Least Mean Squares (LMS) Algorithm
- If e.g. $\eta_t = 1/t$, then

$$\lim_{t\to\infty} P\left(\mathbf{w}(t) = \mathbf{w}^*\right) = 1,$$

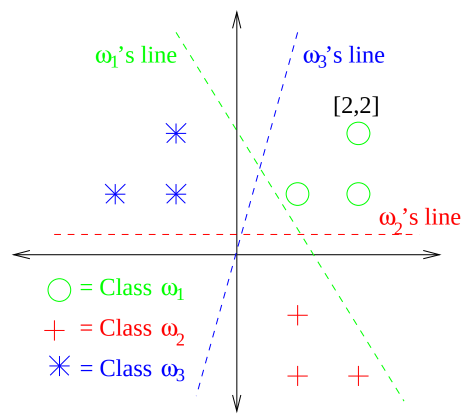
  where

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}\in\Re^\ell}\left\{\mathrm{E}\left[\left|y - \mathbf{w}^T\cdot\mathbf{x}\right|^2\right]\right\}$$

  is vector minimizing mean square error (MSE)

*Note that here $\mathbf{w}(t)$ is weight before trial t. In book it is weight after trial t.

Multiclass learning
Kessler's Construction



- For* $\mathbf{x} = [2,2,1]^T$ of class ω_1, want

$$\sum_{i=1}^{\ell+1} w_{1i}x_i > \sum_{i=1}^{\ell+1} w_{2i}x_i \quad \text{AND} \quad \sum_{i=1}^{\ell+1} w_{1i}x_i > \sum_{i=1}^{\ell+1} w_{3i}x_i$$

*The extra 1 is added so threshold can be placed in $\mathbf{w}$.

Multiclass learning
Kessler's Construction (cont'd)

- So map $\mathbf{x}$ to

$$\mathbf{x}_1 = [\overbrace{2,2,1,}^{\text{orig.}}\overbrace{-2,-2,-1,}^{\text{neg}}\overbrace{0,0,0}^{\text{pad}}]^T$$
$$\mathbf{x}_2 = [2,2,1,0,0,0,-2,-2,-1]^T$$

(all labels $= +1$) and let

$$\mathbf{w} = [\overbrace{w_{11},w_{12},w_{10},}^{\mathbf{w}_1}\overbrace{w_{21},w_{22},w_{20},}^{\mathbf{w}_2}\overbrace{w_{31},w_{32},w_{30}}^{\mathbf{w}_3}]^T$$

- Now if $\mathbf{w}^{*T} \cdot \mathbf{x}_1 > 0$ and $\mathbf{w}^{*T} \cdot \mathbf{x}_2 > 0$, then

$$\sum_{i=1}^{\ell+1} w_{1i}^* x_i > \sum_{i=1}^{\ell+1} w_{2i}^* x_i \quad \text{AND} \quad \sum_{i=1}^{\ell+1} w_{1i}^* x_i > \sum_{i=1}^{\ell+1} w_{3i}^* x_i$$

- In general, map $(\ell+1) \times 1$ feature vector $\mathbf{x}$ to $\mathbf{x}_1, \ldots \mathbf{x}_{M-1}$, each of size $(\ell+1)M \times 1$
- $\mathbf{x} \in \omega_i \Rightarrow \mathbf{x}$ in ith block and $-\mathbf{x}$ in jth block, (rest are 0s). Repeat for all $j \neq i$
- Now train to find weights for new vector space via perceptron, Winnow, etc.

Multiclass learning
Error-Correcting Output Codes (ECOC)

- Since Win. & Percep. learn binary functions, learn individual bits of binary encoding of classes
- E.g. $M = 4$, so use two linear classifiers:

Class	Binary Encoding	
	Classifier 1	Classifier 2
ω_1	0	0
ω_2	0	1
ω_3	1	0
ω_4	1	1

and train simultaneously

- Problem: Sensitive to individual classifier errors, so use a set of encodings per class to improve robustness
- Similar to principle of error-correcting output codes used in communication networks [Dietterich & Bakiri, 1995]
- General-purpose, independent of learner

Topic summary due in 1 week!