Multi-category Generalization: Multi-class Perceptron (DHS 5.12)

Decision Rule:

$g_i(\underline{x})=\underline{w}^{(i)T}\underline{x}$
if $g_i(\underline{x})> g_j(\underline{x})$ for all $j\neq i$, assign $\underline{x}$ to S_i

Algorithm (given in class):
For each prototype $\underline{y}^{(i)}$

If $\underline{y}^{(i)} \in S_i$ but machine assigns it to S_j
Update for S_i, $w^{(i)}(k+1)=\underline{w}^{(i)}(k)+\alpha y^{(i)}$

Update for S_j, $w^{(j)}(k+1)=\underline{w}^{(j)}(k)-\alpha y^{(i)}$

$w^{(l)}(k+1)=\underline{w}^{(l)}(k)$ all $l\neq i,j$ (DHS p. 267, Eq. (115))

If machine classifies $\underline{y}^{(i)}$ correctly, do not update the prototype.

The algorithm is guaranteed to converge (for a fixed increment).

Convergence of algorithm is proved in DHS Sec. 5.12.2 (not covered in class)

Absolute Correction (one example):

If machine puts $\underline{y}_m^{(k)}$ into S_j

Want to:

$$w_k^T(i+1)\underline{y}_m^{(k)} > w_j^T(i+1)\underline{y}_m^{(k)}$$

$$[w_k(i)+\alpha\underline{y}_m^{(k)}]^T\underline{y}_m^{(k)} > [w_j(i)-\alpha\underline{y}_m^{(k)}]^T\underline{y}_m^{(k)}$$

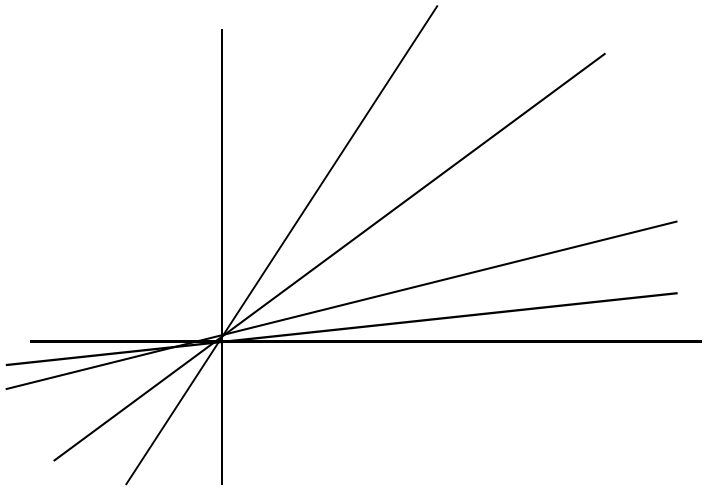
$$w_k^T(i)\underline{y}_m^{(k)} + \alpha\underline{y}_m^{(k)^T}\underline{y}_m^{(k)} > w_j^T(i)\underline{y}_m^{(k)} - \alpha\underline{y}_m^{(k)^T}\underline{y}_m^{(k)}$$

Guarantees re-classification of $\underline{y}_m^{(k)}$ will not result in the same error

$$\alpha = \frac{\left|\underline{w}_j^T(i) - \underline{w}_k^T(i)\right|\underline{y}_m^{(k)}}{2\underline{y}_m^{(k)T}\underline{y}_m^{(k)}}$$

Doesn't guarantee correct classification of $\underline{y}_m^{(k)}$ after 1 update.

Relaxation Procedures (DHS 5.6, p. 235)



Augmented space incorporates a safety margin.
Gives a smoother surface than Perceptron

Perceptron: $J(\underline{w})=\sum_{\underline{y}\in\underline{Y}}(-\underline{w}^T\underline{y})$ (DHS p. 227, Eq. (16))

New cost: $J(\underline{w})=\sum_{\underline{y}\in\underline{Y}}(\underline{w}^T\underline{y})^2$ (DHS p. 235, Eq. (32))

See Fig. 5.11 for the gradients of various $J(\underline{w})$'s

Relaxation

Cost function with margin:

$$J(\underline{w})=\frac{1}{2}\sum_{\underline{y}\in\underline{Y}}\frac{[\underline{w}^T\underline{y}-b]^2}{\left\|\underline{y}\right\|^2}$$ (DHS p. 235, Eq. (33))

b is a margin factor

$\underline{Y}$ represents the set of prototypes for which $\underline{w}^T\underline{y}\le b$
if $\underline{Y}=\varnothing$ then $J(\underline{w})=0$
$J(\underline{w})\ge 0$
$J(\underline{w})=0$ only of $\underline{w}^T\underline{y}\ge b$, $\forall\underline{y}$

The gradient of the cost function with margin:

$$\nabla_{\underline{w}} J(\underline{w}) = \sum \frac{(\underline{w}^T \underline{y} - b)}{\left\|\underline{y}\right\|^2} \underline{y}$$

(Appendix A2.4 for vector matrix calculus)

Gradient Descent:

$\underline{w}(k+1)=\underline{w}(k)-\alpha(k)\nabla J(\underline{w})$

Relaxation Procedure

i) One at a time = Single-sample Relaxation with Margin (DHS, p. 236 Algorithm 9)

$\underline{w}(0)$=arbitrary

$$\underline{w}(k+1)=\underline{w}(k)- \alpha(k) \frac{\underline{w}^T \underline{y}_k - b}{\left\|\underline{y}_k\right\|^2} \underline{y}_k \qquad \text{if } \underline{w}^T(k)\underline{y}_k \le b$$

$\underline{w}(k+1)=\underline{w}(k)$ if $\underline{w}^T(k)\underline{y}_k > b$

ii) Many at a time = Batch Relaxation with Margin (DHS, p. 236 Algorithm 8)

If $\underline{w}(0)$=arbitrary

$$\underline{w}(k+1)=\underline{w}(k)-\alpha(k) \sum \frac{\underline{w}^T \underline{y} - b}{\left\|\underline{y}\right\|^2} \underline{y}$$

Compare to Perceptron Fraction Correction

$$\underline{w}(k+1)=\underline{w}(k)-\lambda\ [\frac{\underline{w}^T \underline{y}_k}{\|\underline{y}_k\|^2}]\underline{y}_k$$

Relaxation – move fraction α of distance from $\underline{w}(k)$ to hyperplane $\underline{w}^T(k)y_k-b=0$

- $\alpha=1$ => move onto hyperplane
- $\alpha<1$ => underrelaxation
- $\alpha>1$ => overrelaxation

Relaxation Rule Convergence

- Linearly separable prototypes
- Either converges to a solution in a finite number of steps
- Or approaches a $\underline{w}$ on the boundary of $\underline{w}^T\underline{y}\geq b$ solution region.
- If $b>0$, at some point $\underline{w}$ will eventually enter $\underline{w}^T\underline{y}>0$ region and remain in that region at infinitum.

So far,

Previous Methods - Error Correcting

- **Update only for misclassified samples**
- **Search for error-free solutions**

Minimum Squared-Error Procedures: Pseudo-inverse (DHS 5.8)

- Again a 2-class problem
- Assume reflected prototypes
- Minimum mean square error (MSE) technique (=> Least Mean Squares, LMS).
- Consider all samples

Pseudo-inverse (continues)

b = arbitrary, will get a solution whether data is separable or not, but no guarantee it is a good solution for separating prototypes.

If b is carefully chosen, we may be able to get a good discriminant function for both separable and non-separable cases.

MSE solution depends on the target vector b

Different choices for b give the solution different properties

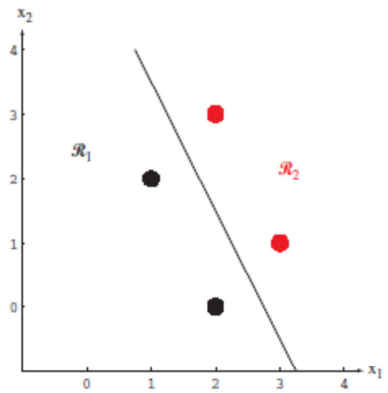
Example 1: Constructing a linear classifier by matrix pseudoinverse

Suppose we have the following two-dimensional points for two categories: ω_1: $(1,2)^t$ and $(2,0)^t$, and ω_2: $(3,1)^t$ and $(2,3)^t$, as shown in black and red, respectively, in the figure.
Our matrix $\mathbf{Y}$ is therefore

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{pmatrix}$$

and after a few simple calculations we find that its pseudoinverse is

$$\mathbf{Y}^\dagger \equiv \lim_{\epsilon \to 0} (\mathbf{Y}^t\mathbf{Y} + \epsilon\mathbf{I})^{-1}\mathbf{Y}^t = \begin{pmatrix} 5/4 & 13/12 & 3/4 & 7/12 \\ -1/2 & -1/6 & -1/2 & -1/6 \\ 0 & -1/3 & 0 & -1/3 \end{pmatrix}$$



Four training points and the decision boundary $\mathbf{a}^t \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} = 0$, where $\mathbf{a}$ was found by means of a pseudoinverse technique.

We arbitrarily let all the margins be equal, i.e., $\mathbf{b} = (1,1,1,1)^t$. Our solution is $\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b} = (11/3, -4/3, -2/3)^t$, and leads to the decision boundary shown in the figure. Other choices for $\mathbf{b}$ would typically lead to different decision boundaries, of course.

Windrow-Hoff (DHS 5.8.4)

Use this cost function, $J(w)=\|Y\underline{w}-\underline{b}\|^2$
$\underline{b}$=target vector

Advantages over pseudoinverse:

- Pseudoinverse can be very large
- It could be singular
- It can have truncation problems, errors.

Here

- One-at-a-time update
- Feedback scheme to reduce truncation errors.

$\nabla J = 2Y^T(\underline{Y}\underline{w}-\underline{b})$

Rule 1=> $\underline{w}(k+1)=\underline{w}(k)+\alpha(k)Y^T(Yw(k)-b)$ for all samples

or

Rule 2=> $\underline{w}(k+1)=\underline{w}(k)+\alpha(k)[\underline{b}(k)-\underline{w}^T(k)y_k]$ y_k considering the samples sequencially

$\underline{w}(0)$=arbitrary

The size of Y^TY is smaller than Y^+, storage requirements are less.
Update for all prototypes (misclassified & correctly-classified)
Usually updates never cease
$\alpha(k)=\alpha(0)/k$ for convergence.
Requires a good b.

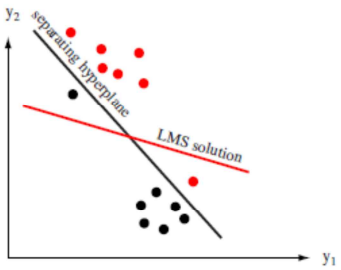


Figure 5.17: The LMS algorithm need not converge to a separating hyperplane, even if one exists. Since the LMS solution minimizes the sum of the squares of the distances of the training points to the hyperplane, for this exmple the plane is rotated clockwise compared to a separating hyperplane.