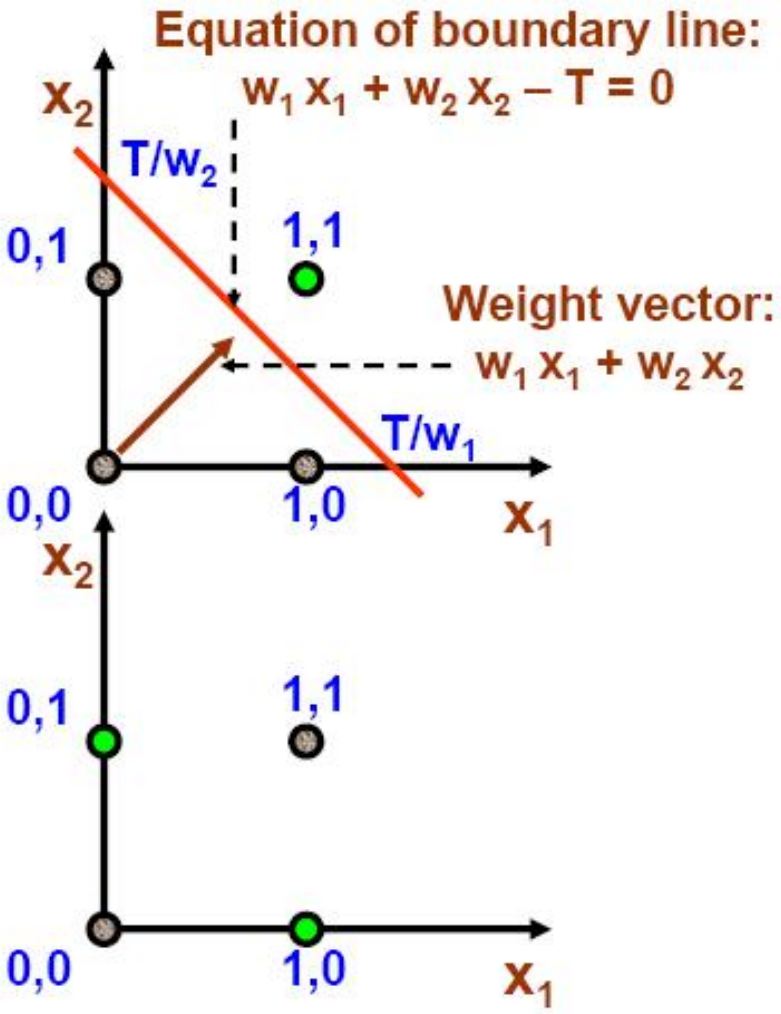
Chapter 5:
Linear Discriminant Functions
(Section 5.4)

- Two-Category Linearly Separable Case
- Gradient Descent
- Newton's Algorithm

SEPARATING PATTERNS

- **AND Gate:** patterns can be separated by a single line (they are said to be **linearly separable**). Same logic is applicable to an **OR-gate**.
- **XOR Gate:** patterns cannot be separated by a single line (they are said to be **non-linearly separable**).



LINEAR SEPARATORS

- **Definition:** two sets of pattern samples are said to be **linearly separable** if there exists a linear discriminant function that separates the sets while classifying all the samples correctly.

- **LDF forms:**
  - $g(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_dx_d \geq 0$
  - $g(x) = w_1x_1 + w_2x_2 + \ldots + w_dx_d \geq \theta$

- **Main problems:**
  - Finding the discriminant that accurately separates two sets of linearly separable patterns.
    - Method: **gradient descent**.
  - Optimize the weights to minimize the error, that is the measure of how much the predictions deviate for the desired answers.
    - Method: **linear regression optimization**.

Training Patterns : y1,…,yn

a: weights

$g(\mathbf{x})=\mathbf{a}^t\mathbf{y}$ $\mathbf{a}^t\mathbf{y}_i>0$, $\mathbf{y}_i$ belongs to ω_1

$\mathbf{a}^t\mathbf{y}_i<0$, $\mathbf{y}_i$ belongs to ω_2

Training Patterns : y1,…,yn

a: weights

Replace all ys belonging to ω_2 by their negatives

So $g(\mathbf{x})=\mathbf{a}^t\mathbf{y}$ $\mathbf{a}^t\mathbf{y}_i>0$

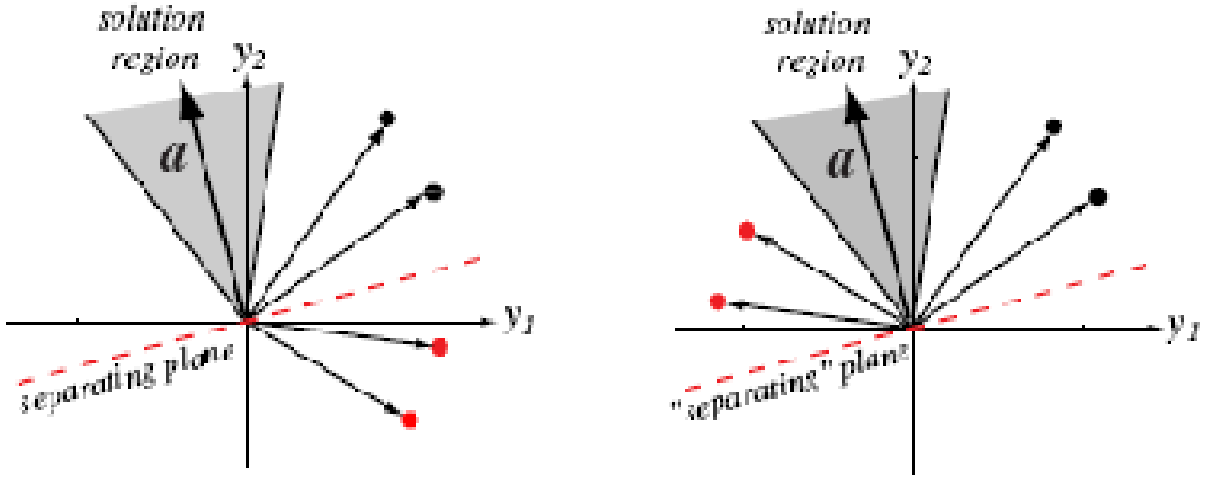


FIGURE 5.8. Four training samples (black for ω_1, red for ω_2) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been "normalized"—that is, changed in sign. Now the solution vector leads to a plane that places all "normalized" points on the same side. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Training Patterns : y1,y2 and y3

a: weights

g(**x**)=$\mathbf{a}^t\mathbf{y}$ $\mathbf{a}^t\mathbf{y}_i \geq b > 0$, b is a positive constant called margin

-Here we exchange a and y. In the space of a1 and a2 we search for the solution region that satisfies $a^t y > 0$ (the left)

-In the right we consider each y as W and b as W0. ($y^t a - b > 0$)

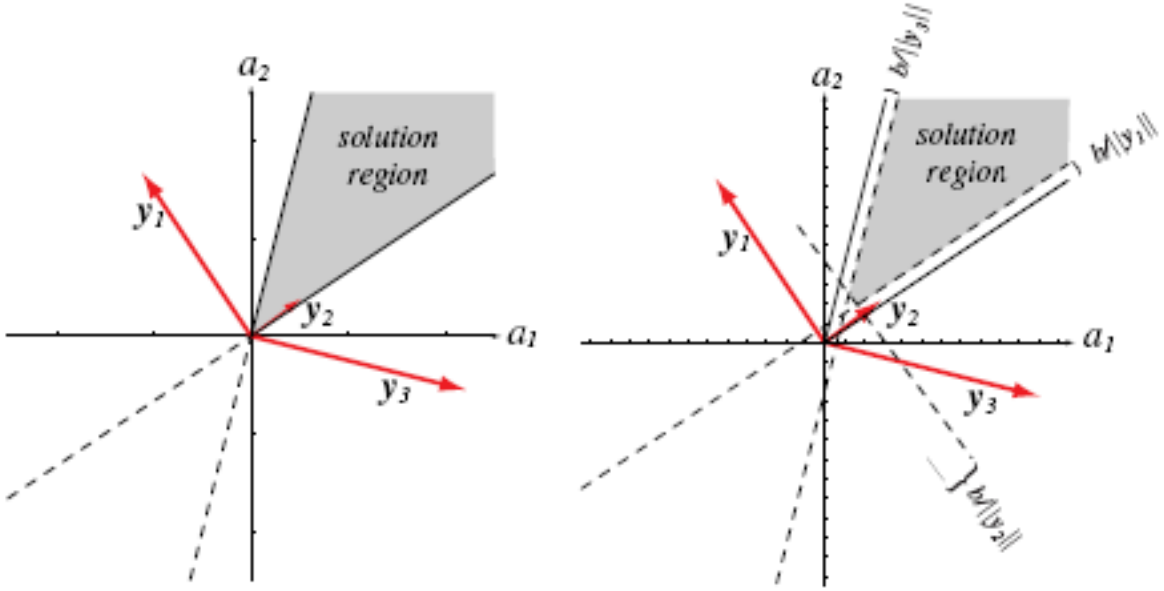
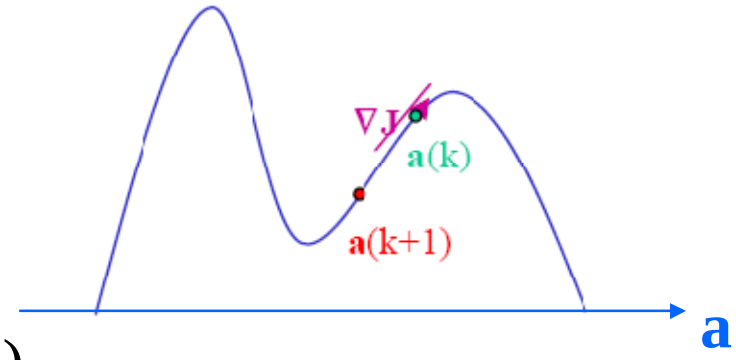


FIGURE 5.9. The effect of the margin on the solution region. At the left is the case of no margin ($b = 0$) equivalent to a case such as shown at the left in Fig. 5.8. At the right is the case $b > 0$, shrinking the solution region by margins $b/\|y_i\|$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

How to find the solution for **a**

Using the Gradient Descent concept

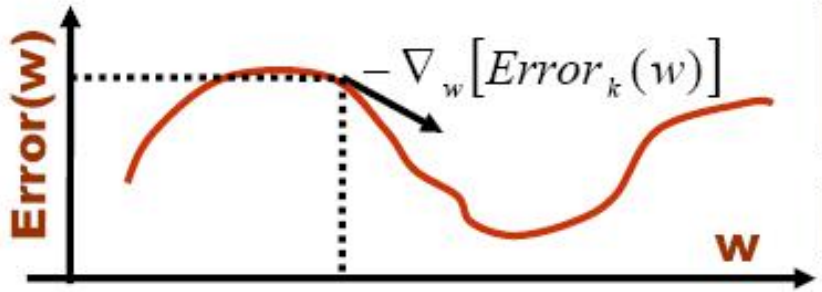


$$a(k+1) = a(k) - h(k)\nabla J(a(k))$$

$$a(k+1) = a(k) - b(k)\nabla J(a(k))$$

GRADIENT DESCENT METHOD

- **Principle:** optimize the weights by descending towards the minimum of the error function guided by the gradient information.
- **Method:** steepest descent is achieved by changing the weights after every sample according to the gradient rule (also known as the **delta rule**) that recommends following the direction opposite to the error gradient.



$$w_j \leftarrow w_j - \beta \frac{\partial}{\partial w_j} [Error_k(w)]$$

$$w \leftarrow w - \beta \nabla_w [Error_k(w)]$$

- The step size β (where $\beta > 0$) is called the **learning rate**.

LEARNING RATE

- The size of the step used for adjusting the weights while training a LDF-based classifier is called the **learning rate**.

- If learning rate is represented by β, then $\beta > 0$. Usually: $0 < \beta < 1$.
- The choice for the learning rate is critical:
  - If β is too small, then the convergence might be needlessly slow.
  - If β is too large, the convergence might overshoot (and miss the minimum).

- **Design alternatives:**
  - **Fixed-increment case:** β is constant throughout the training session.
  - **Variable-increment case:** β is not constant (it is usually decreased as the training progresses) .

- *A* typical choice for the variable-increment learning rate is:

$$\beta(k) = 1/k \text{ or } \beta(k) = \beta(1)/k$$

where k is the index of current training pass, and $\beta(1)$ a positive constant.

BASIC GRADIENT DESCENT

Gradient Descent Procedure

begin

set

θ = error threshold.

$\beta(1)$ = learning rate

P = maximum number of passes.

initialize weights $w = \{w_0, w_1, \ldots, w_d\}$

$k = 0$

do

$k \leftarrow k+ 1$

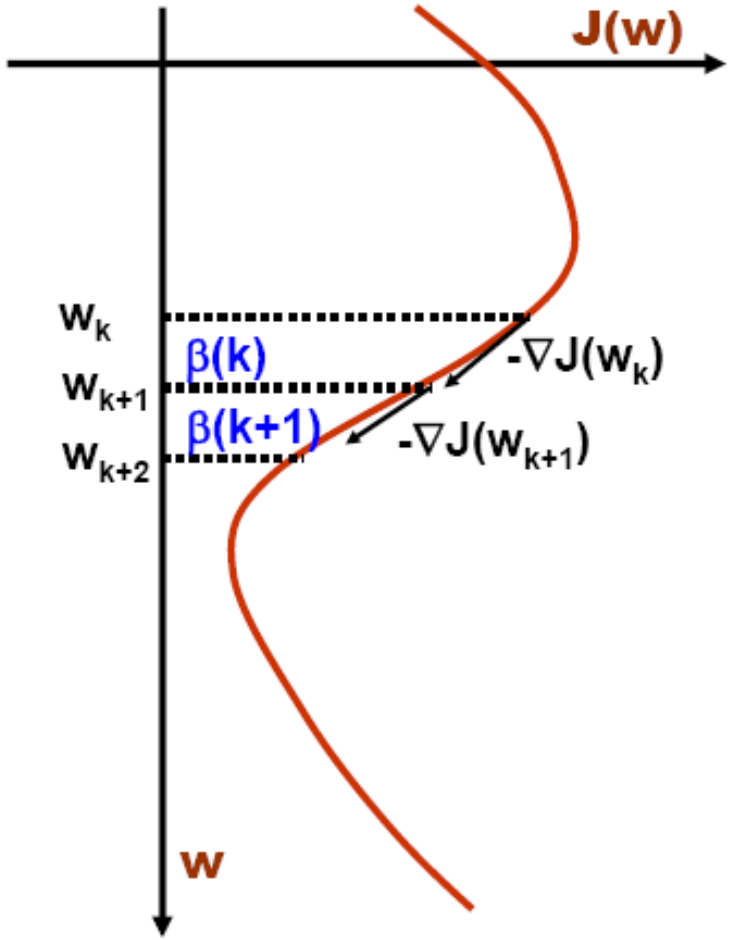
$\varepsilon \leftarrow \beta(k) \nabla J(w)$

$w \leftarrow w - \varepsilon$

until $((|\varepsilon| \leq \theta)$ or $(k \geq P))$

return (w)

end



TWO-CATEGORY EXAMPLE

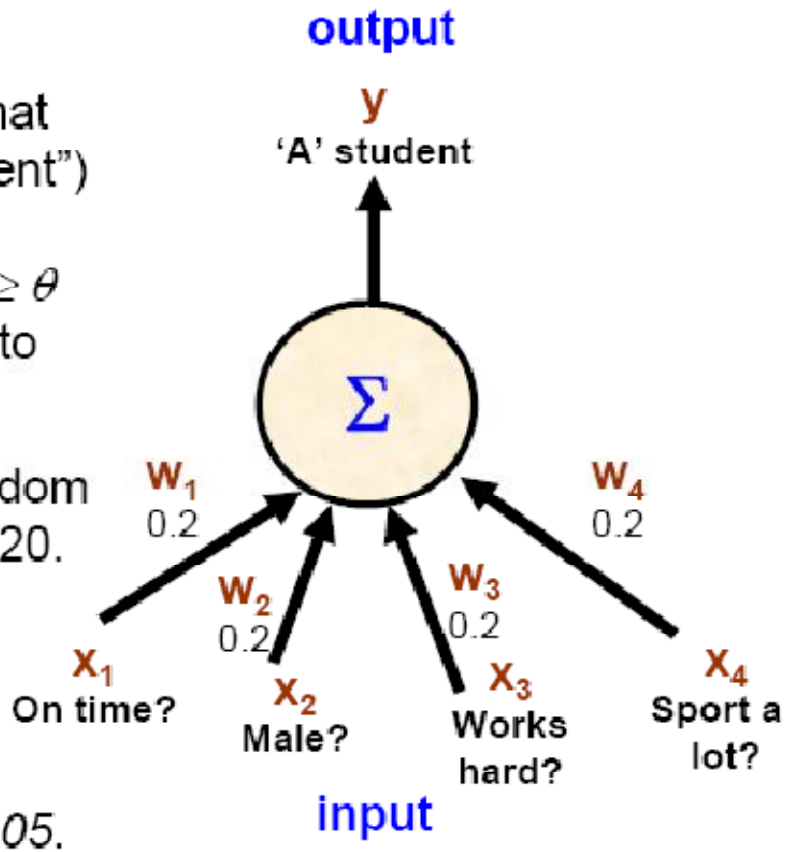
Patterns	Input (Features)				Output
Student	On time?	Male?	Works hard?	Sport a lot?	'A' Mark
Richard	Yes	Yes	No	Yes	No
Alan	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	No
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No

CLASSIFIER SETUP

- Assume a threshold $\theta = 0.6$.
- Classifier output is evaluated to 1 (that is, assumed to represent an "A student") if the LDF yields a value $y \geq \theta$, or:

  $g(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \geq \theta$

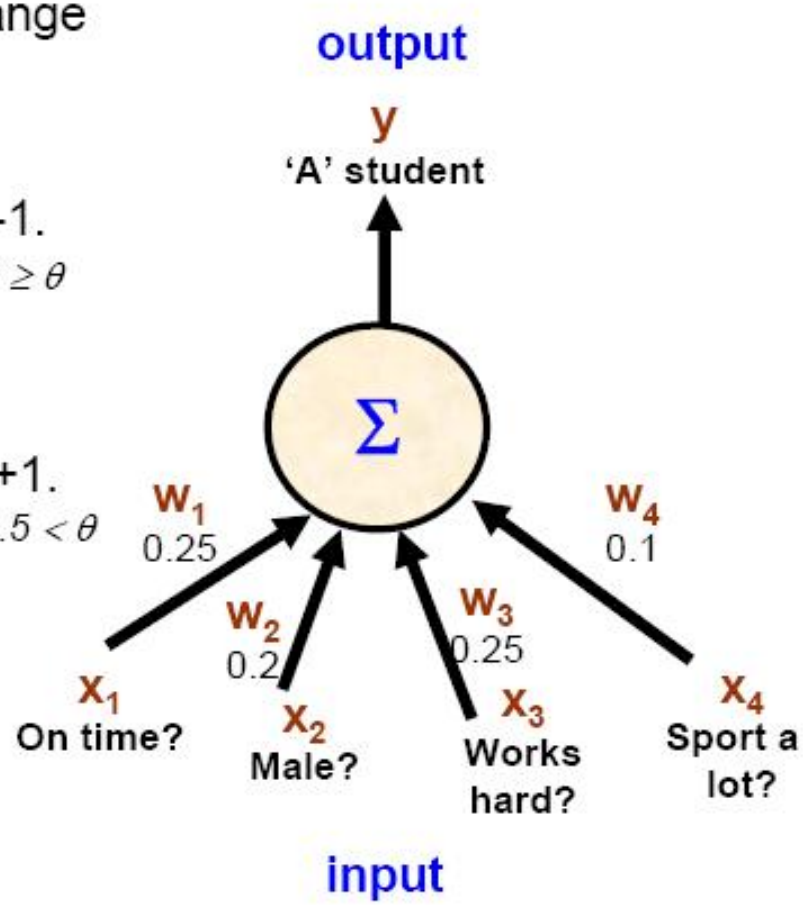
  Otherwise, the output is considered to be 0 (i.e. not an "A student").
- All the weights are initially set to random values; here all weights are set to 0.20.



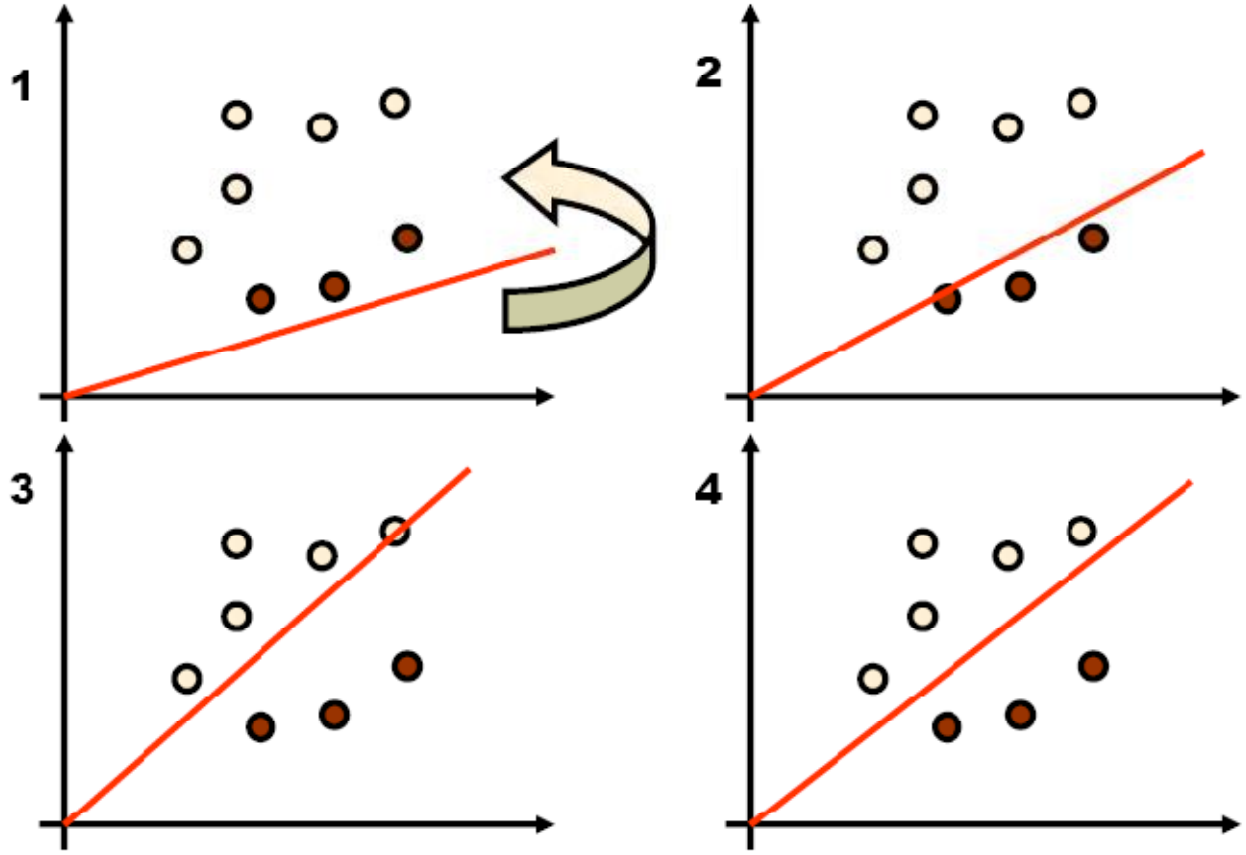
- The choice for learning rate is $\beta = 0.05$.

GRADIENT DESCENT TRAINING

- **Observation:** for this case the learning rate is approximating the weight change (increment or decrement).
- Richard:
  - Output: 1 (instead of 0). Error = -1.
    $(1 \times 0.2 + 1 \times 0.2 + 0 \times 0.2 + 1 \times 0.2) = 0.6 \geq \theta$
  - Reduce active weights by 0.05.
- Alan:
  - Output: 0 (instead of 1). Error = +1.
    $(1 \times 0.15 + 1 \times 0.15 + 1 \times 0.2 + 0 \times 0.15) = 0.5 < \theta$
  - Increase active weights by 0.05.
- Alison … Jeff … Gail ... Simon …
- Weights when training is complete: 0.25, 0.2, 0.25, 0.1



REGRESSION LEARNING



How to set the learning rate

Criterion function (J) can be approximated by:

$$J(a(k+1)) \approx J(a(k)) + \nabla J^t (a(k+1) - a(k)) + \frac{1}{2}(a(k+1) - a(k))^t H (a(k+1) - a(k))$$

H: Hessian matrix

Example of two-dimensional case:

$$H = \begin{bmatrix} \frac{\partial^2 J}{\partial^2 a_1} & \frac{\partial^2 J}{\partial a_1 \partial a_2} \\ \frac{\partial^2 J}{\partial a_2 \partial a_1} & \frac{\partial^2 J}{\partial^2 a_2} \end{bmatrix}$$

Using

$$a(k+1) = a(k) - h(k)\nabla J(a(k))$$

We have

$$J(a(k+1)) \approx J(a(k)) - h(k)\|\nabla J\|^2 + \frac{1}{2}h^2(k)\nabla J^t H \nabla J$$

It can be minimized by:

$$h(k) = \frac{\|\nabla J\|^2}{\nabla J^t H \nabla J}$$

NEWTON DESCENT

- **Principle:** approximates the classification error with the second order expansion (i.e. using second partial derivatives) of *J(w)*.

- **Method:** adjusts the weights to minimize the second error expansion by adopting a dynamic learning rate based on the **Hessian matrix**.

$$w \leftarrow w - H^{-1} \nabla J(w)$$

where *H* is the *Hessian matrix* of second partial derivatives $\partial^2 J(w)/\partial w_i \partial w_j$.

- **Advantage:** Newton's method (black line) typically leads to greater improvements per step than basic gradient descent method (red line).

- **Disadvantage:** added computational burden for calculating and inverting the Hessian matrix (not always justified).