Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava

Neural Networks

Lecture 2

Simple perceptron

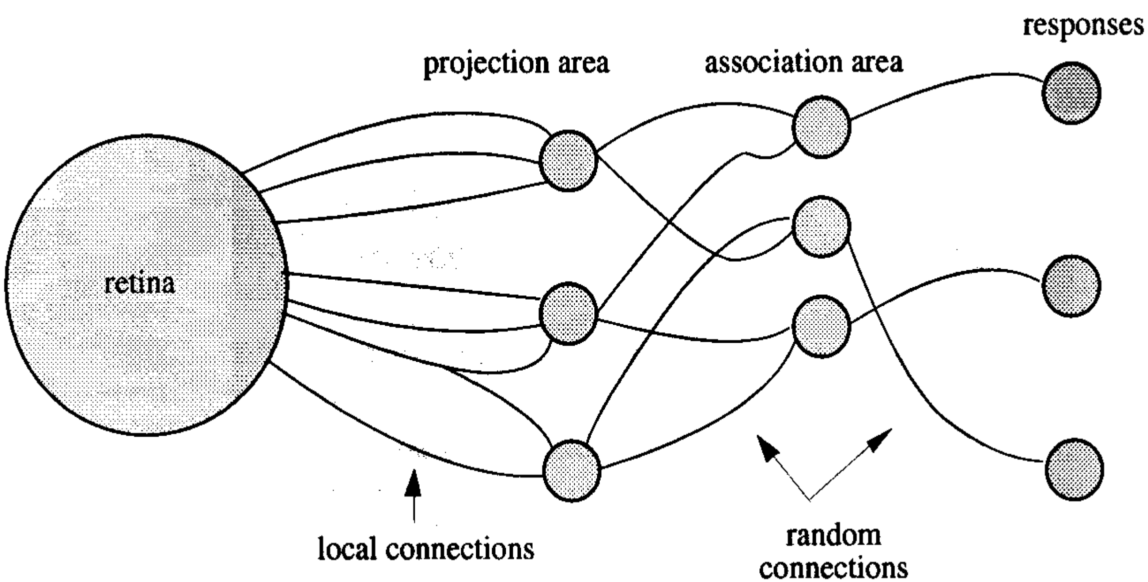
Igor Farkaš

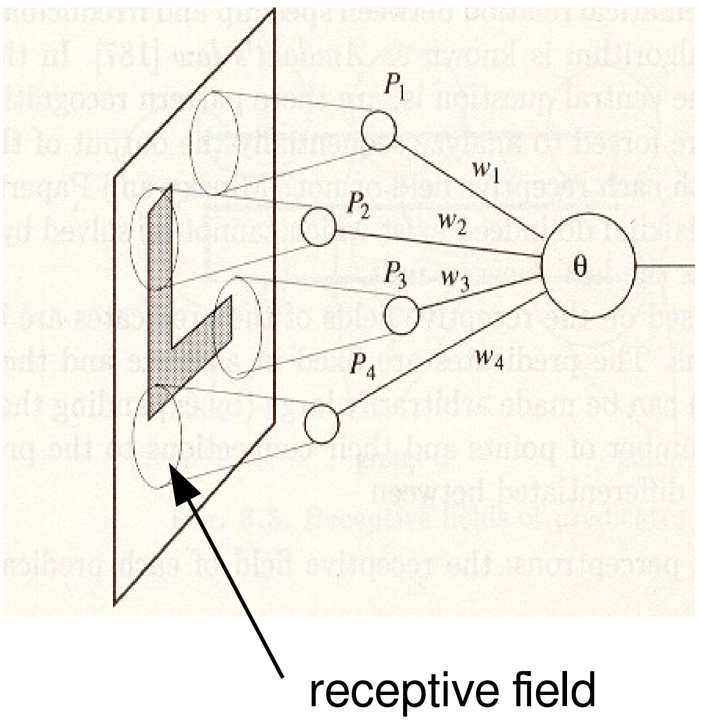
2024

Classical perceptron

In 1958, F. Rosenblatt (American psychologist) proposed perceptron, a more general computational model (than McCulloch-Pitts' TL units) with free parameters, stochastic connectivity and threshold element.

In 1950, Hubel & Wiesel "decoded" the structure of retina and receptive fields.





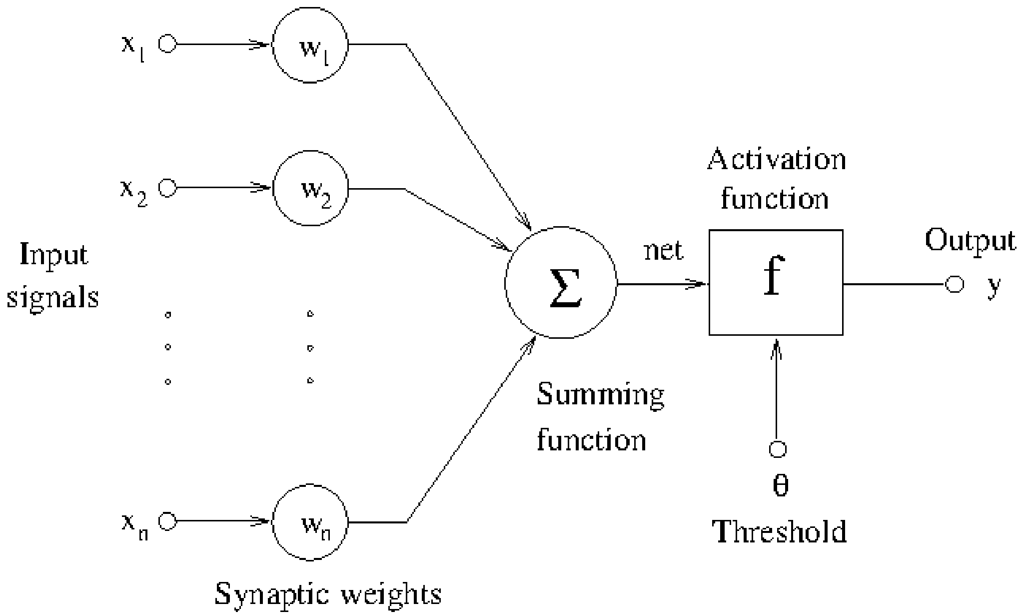
Discrete perceptron

- Inputs $\boldsymbol{x}$, weights $\boldsymbol{w}$, output y
- Function:

$$y = f\left(\sum_{j=1}^{n} w_j x_j - \theta\right)$$

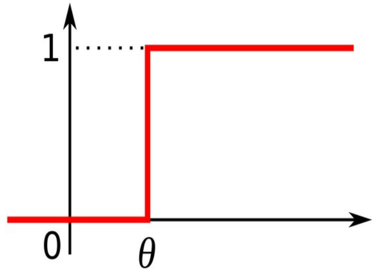
$$y = f\left(\sum_{j=1}^{n+1} w_j x_j\right) \qquad x_{n+1} = -1$$

$$w_{n+1} = \theta$$



- f = threshold (step) function
- Supervised learning – with teacher d
- e.g. $d = 1$ for $\forall \boldsymbol{x} \in$ C1, $d = 0$ for $\forall \boldsymbol{x} \in$ C2
- Learning: if $\boldsymbol{w}^T\boldsymbol{x} \leq 0$ but $\boldsymbol{x} \in$ C1, then $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \boldsymbol{x}$

  if $\boldsymbol{w}^T\boldsymbol{x} > 0$ but $\boldsymbol{x} \in$ C2, then $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \boldsymbol{x}$



Or: $w_j(t+1) = w_j(t) + \alpha\ (d - y)\ x_j$ $\qquad \alpha$ = learning rate

Rosenblatt F. (1962). *Principles of Neurodynamics*, Spartan, New York.

Perceptron algorithm

Given: input-target $\{\boldsymbol{x}^{(p)}, d^{(p)}\}$ pairs, unipolar perceptron

Initialization: randomize weights, shuffle the pairs, set learning rate, set $E = 0$.

Training:

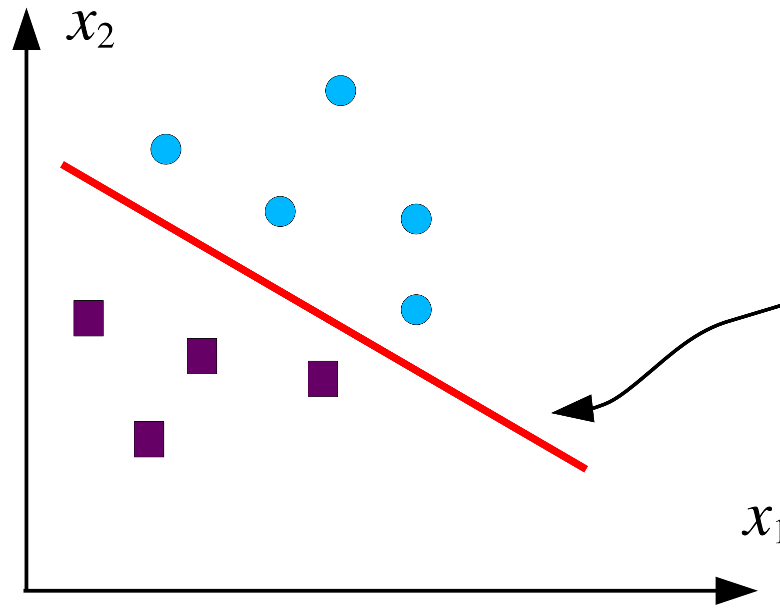
1. choose next input $\boldsymbol{x}$, compute output y
2. evaluate error function, $e(t) = ½\,(d^{(p)} - y^{(p)})^2$, $E = E + e(t)$
3. if $e(t) > 0$, adjust weights using perceptron rule
4. if not all inputs used, then goto 1, else goto 5
5. if $E == 0$ (all inputs in the set classified correctly), then end else reshuffle the pairs, $E = 0$, go to 1.

Perceptron classification capacity

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = \theta$$

linear separability of two classes

2D case



linear decision boundary

Fixed-increment convergence theorem (Rosenblatt, 1962): *"Let the classes A and B are finite and linearly separable, then perceptron learning algorithm converges (updates its weight vector) in a finite number of steps."*

Finding a solution

$\boldsymbol{w}^T\boldsymbol{x} > 0$ for $C1$

$\boldsymbol{w}^T\boldsymbol{x} \leq 0$ for $C2$

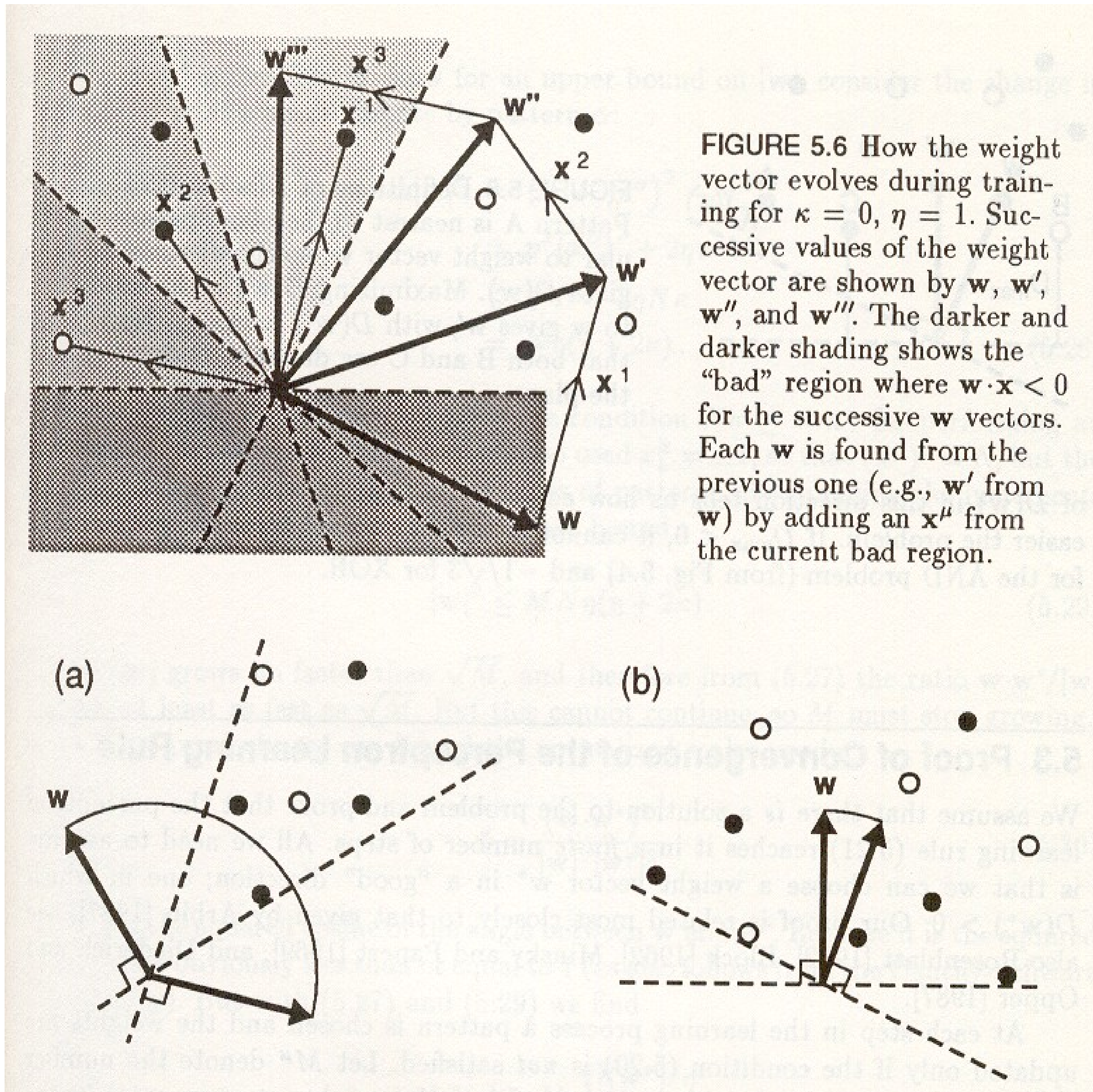


FIGURE 5.6 How the weight vector evolves during training for $\kappa = 0$, $\eta = 1$. Successive values of the weight vector are shown by $\mathbf{w}$, $\mathbf{w}'$, $\mathbf{w}''$, and $\mathbf{w}'''$. The darker and darker shading shows the "bad" region where $\mathbf{w}\cdot\mathbf{x} < 0$ for the successive $\mathbf{w}$ vectors. Each $\mathbf{w}$ is found from the previous one (e.g., $\mathbf{w}'$ from $\mathbf{w}$) by adding an $\mathbf{x}^\mu$ from the current bad region.

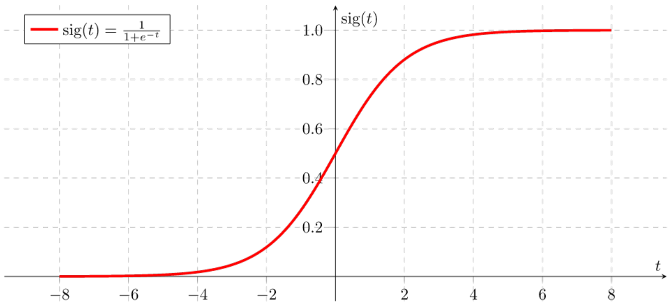
easy

more difficult

(Hertz et al, 1990)

Continuous perceptron

- Nonlinear unit with activation function: $y = f(net) = 1 \,/\, (1+e^{-net})$
- Has nice properties:
  - boundedness
  - monotonicity
  - differentiability



(unipolar) sigmoid

- Error function (e.g. quadratic): $E(\boldsymbol{w}) = \Sigma_p\, e^{(p)} = ½\, \Sigma_p (d^{(p)} - y^{(p)})^2$ over dataset p , also called loss function (objective function)
- We want to minimize the error function: necessary conditions $e(\mathbf{w}^*) \le e(\mathbf{w})$ and $\nabla e(\boldsymbol{w}^*) = 0$, gradient operator

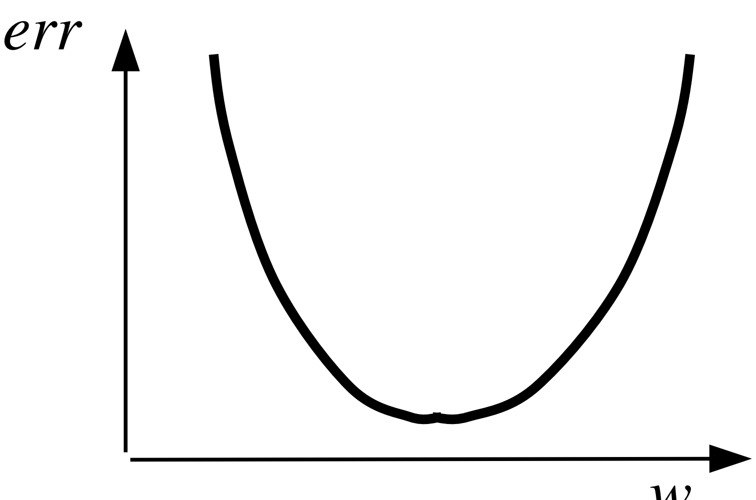
  $\nabla = [\partial/\partial w_1, \partial/\partial w_2, \ldots]^T$. Minimizing $E(\boldsymbol{w})$ leads to
- (stochastic, online) gradient descent learning:

$$w_j(t+1) = w_j(t) + \alpha\ (d^{(p)} - y^{(p)})\, f\,'(net)\, x_j \;=\; w_j(t) + \alpha \delta^{(p)} x_j^{(p)}$$

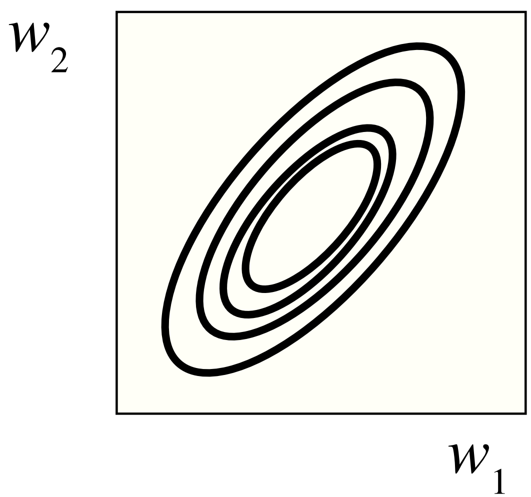
Error surface for a continuous perceptron

- Assume 1 neuron, linear or with a sigmoid function
- The output error $e = f(w_1, w_2, \ldots, w_n)$, assume quadratic error
- For a linear neuron with *n* inputs, we have a convex function (quadratic bowl); vertical cross-sections are parabolas; horizontal cross-sections are ellipses.

err = 3rd dim.



1-dim.

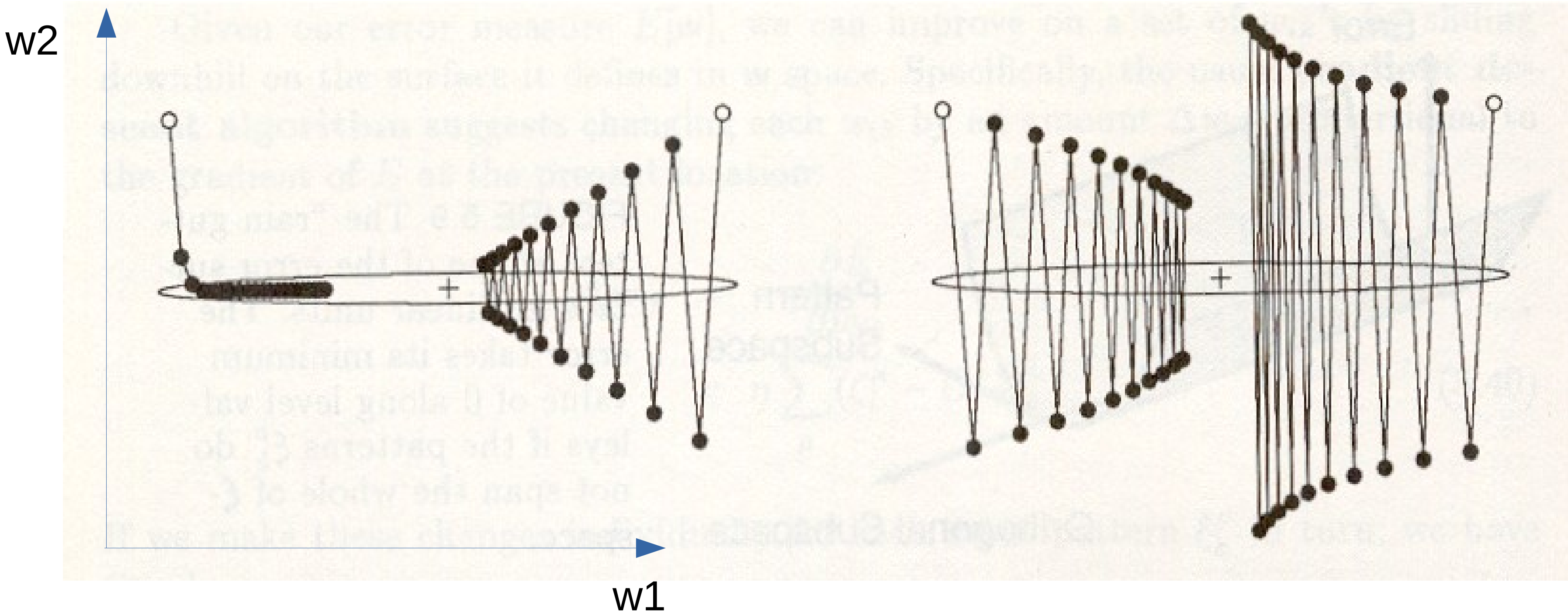


2-dim.

weight space

Effect of learning rate

$$E = w_1^2 + 20\, w_2^2$$



0.02 0.0476 0.049 0.0505

(Hertz et al, 1990)

Linear neuron as a least-squares filter

- Consider: $y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{w},$ input-target pairs $\{\boldsymbol{x}^{(p)}, d^{(p)}\}$, $p = 1, ..., N$
- Collect inputs $\mathbf{X} = [\boldsymbol{x}^{(1)}\, \boldsymbol{x}^{(2)} \ldots.\, \boldsymbol{x}^{(N)}]^{\mathrm{T}}$ ($N\times n$ matrix)
- Let $\boldsymbol{e} = [e^{(1)}\, e^{(2)} \ldots e^{(N)}]^{\mathrm{T}}$ then output error $\boldsymbol{e} = \boldsymbol{d} - \mathbf{X}.\boldsymbol{w}$
- Gauss-Newton method: $E(\boldsymbol{w}) = ½\, \Sigma_p(d^{(p)} - y^{(p)})^2$, compute $\nabla\boldsymbol{e}$ $(n\times N)$
- $j_{pk} = \partial e^{(p)}/\partial w_k \Rightarrow$ Jacobian $\mathbf{J}(t) = [j_{pk}]$ is $(N\times n)$ $\mathbf{J}(t) = -\mathbf{X}(t) = [\boldsymbol{\nabla e}^{\mathrm{T}}]$
- $\boldsymbol{e}'(\boldsymbol{w})^{(N)} = \boldsymbol{e}(\boldsymbol{w}) + \mathbf{J}^{(N)}.(\boldsymbol{w} - \boldsymbol{w}^{(N)})$. [linearity assumption of error f.]
- Substitute $[N \equiv t]$ $\boldsymbol{w}(t+1) = \arg\min_{\boldsymbol{w}} \{1/2\, \|\boldsymbol{e}'(t,\boldsymbol{w})\|^2\}$ …
- Update $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - (\mathbf{J}^{\mathrm{T}}(t)\,\mathbf{J}(t)\,)^{-1}\mathbf{J}(t)\,\boldsymbol{e}(t) = \boldsymbol{w}(t) + (\mathbf{X}^{\mathrm{T}}(t)\,\mathbf{X}(t)\,)^{-1}\mathbf{X}(t)[\,\boldsymbol{d}(t) - \mathbf{X}(t)\,\boldsymbol{w}(t)] = [\mathbf{X}^{\mathrm{T}}(t)\,\mathbf{X}(t)\,)^{-1}\mathbf{X}(t)]\,\boldsymbol{d}(t) \Rightarrow \boldsymbol{w}(t+1) = \mathbf{X}^{+}(t)\,\boldsymbol{d}(t)$
- *"The weight vector* $\boldsymbol{w}(t+1)$ *solves the least-squares problem in an observation interval until time t."*
- neuron as a linear regressor

Alternative loss function: cross entropy

- Useful for classification, leads to probability (uncertainty)
- Error function – cross-entropy (for one output):

  [relative entropy b/w empirical probability distribution $(d^{(p)}, 1 - d^{(p)})$ and output distribution $(y, 1 - y)$]

$$E_{CE}(\boldsymbol{w}) = \Sigma_p E^{(p)} = -\Sigma_p [d^{(p)} \ln y^{(p)} + (1 - d^{(p)}) \ln (1 - y^{(p)})]$$

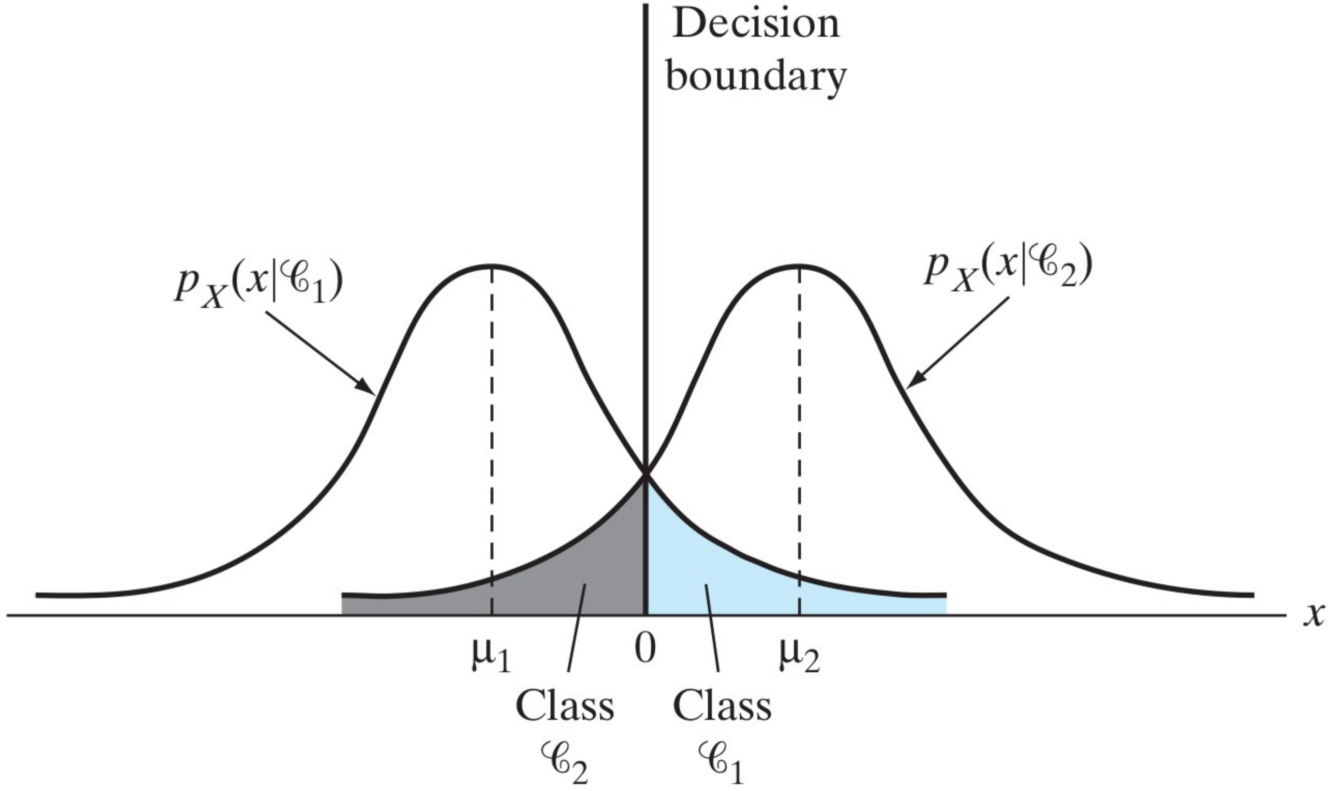
- $\rightarrow$ negative log-likelihood (in probabilistic models)
- minimization of $E^{(p)}$ results in a learning rule:

$$w_j(t+1) = w_j(t) + \alpha \ (d^{(p)} - y^{(p)}) \ x_j^{(p)}$$

- *Note*: In case of 2 classes one can use logistic unit (for C_1: $d = 1$; C_2: $d = 0$),
- then the output y can be interpreted as $P(\mathrm{C}_1 | \boldsymbol{x}) = 1 - P(\mathrm{C}_2 | \boldsymbol{x})$

Bayes classifier for two classes

- (linear) Bayes classifier for a 1D Gaussian environment
- for convenience, decision border is at x = 0



Perceptron link to Bayes classifier

Assumptions:

- random vector $\boldsymbol{X}$, two classes C_1: $E[\boldsymbol{X}] = \boldsymbol{m}_1$, C_2: $E[\boldsymbol{X}] = \boldsymbol{m}_2$
- covariance matrix $\mathbf{C} = E[(\boldsymbol{X} - \boldsymbol{m}_1)(\boldsymbol{X} - \boldsymbol{m}_1)^{\mathrm{T}}] = E[(\boldsymbol{X} - \boldsymbol{m}_2)(\boldsymbol{X} - \boldsymbol{m}_2)^{\mathrm{T}}]$

  We can express conditional probability density function:

  $$f(\boldsymbol{x}|\mathrm{C}_i) = [(2\pi)^{m/2}\det(\mathbf{C})^{1/2}]^{-1} \exp[-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^{\mathrm{T}}\mathbf{C}^{-1}(\boldsymbol{x} - \boldsymbol{m}_i)]$$

  $\boldsymbol{x}$ – observation vector, $i = \{1,2\}$

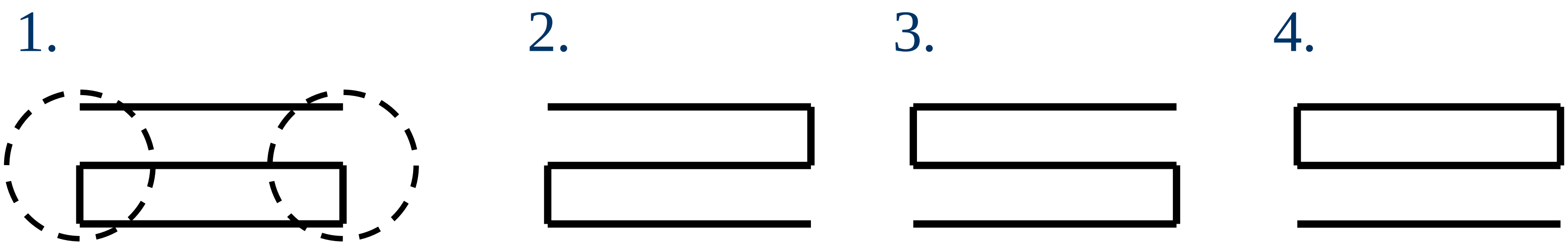
- the 2 classes are equiprobable, i.e. $p_1 = p_2$ (a priori probs)
- (mis)classifications carry the same cost, i.e. $\omega_{12} = \omega_{21}$, $\omega_{11} = \omega_{22} = 0$

  Bayes classifier: "*If* $p_1(\omega_{21} - \omega_{11}) f(\boldsymbol{x}|C_1) > p_2(\omega_{12} - \omega_{22}) f(\boldsymbol{x}|C_2)$, *assign the observation vector* $\boldsymbol{x}$ *to* C_1*. Otherwise, assign it to* C_2."

Bayes classifier (ctd)

- Define likelihood ratio $\Lambda(\boldsymbol{x}) = f(\boldsymbol{x} \mid C_1) / f(\boldsymbol{x} \mid C_2)$ and threshold $\xi = [p_2(\omega_{12} - \omega_{22})] / [p_1(\omega_{21} - \omega_{11})]$. Then:
- $\log \Lambda(\boldsymbol{x}) = -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{m}_1)^{\mathsf{T}} \mathbf{C}^{-1}(\boldsymbol{x} - \boldsymbol{m}_1) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_2)^{\mathsf{T}} \mathbf{C}^{-1}(\boldsymbol{x} - \boldsymbol{m}_2)$
- $\log \xi = 0$
- *Then*: we get a linear Bayes classifier $y = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x} + b$ where
  $y = \log \Lambda(\boldsymbol{x}), \quad \boldsymbol{w} = \mathbf{C}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2), \quad b = \frac{1}{2} (\boldsymbol{m}_2^{\mathsf{T}} \mathbf{C}^{-1} \boldsymbol{m}_2 - \boldsymbol{m}_1^{\mathsf{T}} \mathbf{C}^{-1} \boldsymbol{m}_1) \rightarrow$
  log-likelihood test: *If* $y > 0$, *then* $\boldsymbol{x} \in C_1$, *else* C_2.
- Differences b/w Perceptron (P) and Bayes classifier (BC):
  - P assumes linear separability, BC does not
  - P convergence algorithm is non-parametric, unlike BC
  - P convergence algorithm is adaptive and simple, unlike BC.

Perceptron limits - XOR



• Consider a perceptron classifying shapes as connected or disconnected and taking inputs from shape ends (shown as dashed circles for pattern 1)

• The problem arises because a single layer of processing local knowledge cannot be combined into global knowledge

• No feature-weighing machine (such as a simple perceptron) can do this type of separation, because information about the relation between the bits of evidence is lost (proven by Minsky & Papert, 1969)

• This problem caused the loss of interest in connectionism (in 1970s), since many real problems are not linearly separable.

Summary

- binary and continuous perceptron
- single perceptron can linearly separate two classes:
- perceptron as a detector (of half input space)
- optimization: gradient descent learning
- link to adaptive filtering – error correction learning
- two types of error (loss) functions
- link to statistics: probabilistic Bayes classifier
- limitations of a simple perceptron
- a single neuron model, with any activation function, is linear in its parameters.