

# Idegrendszeri modellezés

Orbán Gergő

<http://golab.wigner.mta.hu>

# Neurons and neural networks I.

# High-level vs. low level modeling

Previously in this course:

- \* Mechanistic description of neurons
- \* Descriptive models of neuronal operations

Goals of the next classes:

Are these models compatible with the neural architecture?  
How can neural computations interpreted in this framework?

# Outline

- single neuron
- neural network
  
- supervised learning
- unsupervised learning

# Neural networks, basic concepts

Goal of the neural network can be phrased:  
learning about its input  
forming memories

## Address based memory

- not associative
- not fault tolerant
- not distributed

## Biological memory

- associative
- error tolerant
  - cue error
  - hardware fault
- parallel and distributed

# Terminology

**Architecture:** what variables are involved, how they interact  
(weights, activities)

**Activity rule:** short time-scale behavior  
how activity of neurons depends on inputs

**Learning rule:** long time-scale behavior  
how weights change as a function of activities of  
other neurons, time, or other factors

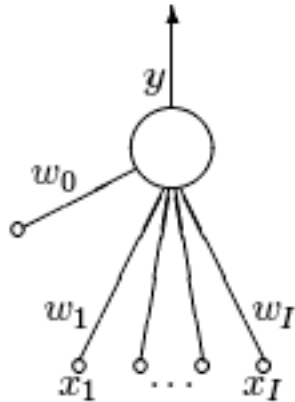
# Terminology

**Supervised learning:** *labelled data* is provided, i.e.  
an input-output relationship is given;  
the ‘teacher’ tells whether the response of  
the neuron to a given input was correct

**Unsupervised learning:** *unlabelled data* is available;  
learns the statistics of input:  
generating a table of occurrences  
some more sophisticated *representation*  
is learned

# Single neuron as a classifier

## Definitions



**Architecture:**  $I$  inputs  $x_i$   
weights  $w_i$

**Activity rule:**

1. activation is calculated

$$a = \sum_i w_i x_i,$$

2. output is calculated

- linear  $y(a) = a.$
- logistic

- tanh  $y(a) = \frac{1}{1 + e^{-a}}$

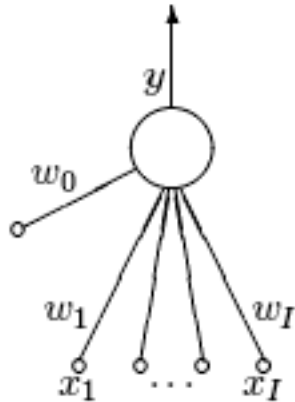
- threshold  $y(a) = \tanh(a)$

$$y(a) = \Theta(a) \equiv \begin{cases} 1 & a > 0 \\ -1 & a \leq 0. \end{cases}$$



# Single neuron as a classifier

## Definitions



**Architecture:**  $I$  inputs  $x_i$   
weights  $w_i$

**Activity rule:**

1. activation is calculated

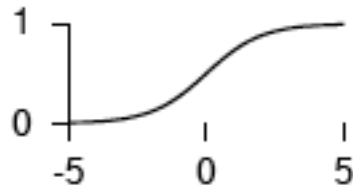
$$a = \sum_i w_i x_i,$$

2. output is calculated

- linear  $y(a) = a.$
- logistic
- tanh
- threshold

$$y(a) = \frac{1}{1 + e^{-a}}$$

$$y(a) = \tanh(a)$$



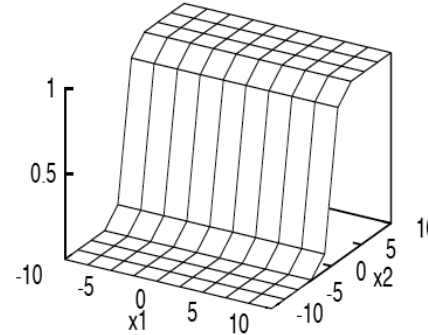
$$y(a) = \Theta(a) \equiv \begin{cases} 1 & a > 0 \\ -1 & a \leq 0. \end{cases}$$

# Single neuron as a classifier

## Concepts

**Input space**

$$y(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}}$$

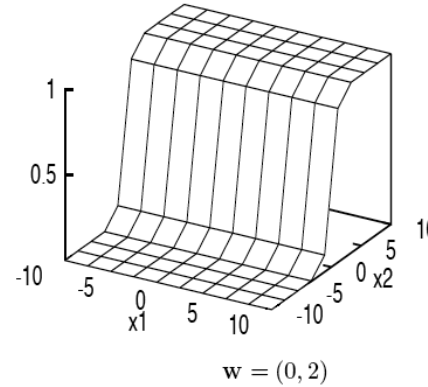


# Single neuron as a classifier

## Concepts

**Input space**

$$y(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}}$$

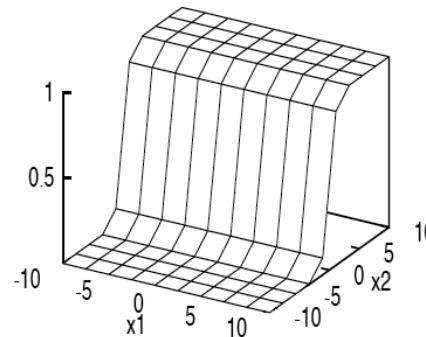


# Single neuron as a classifier

## Concepts

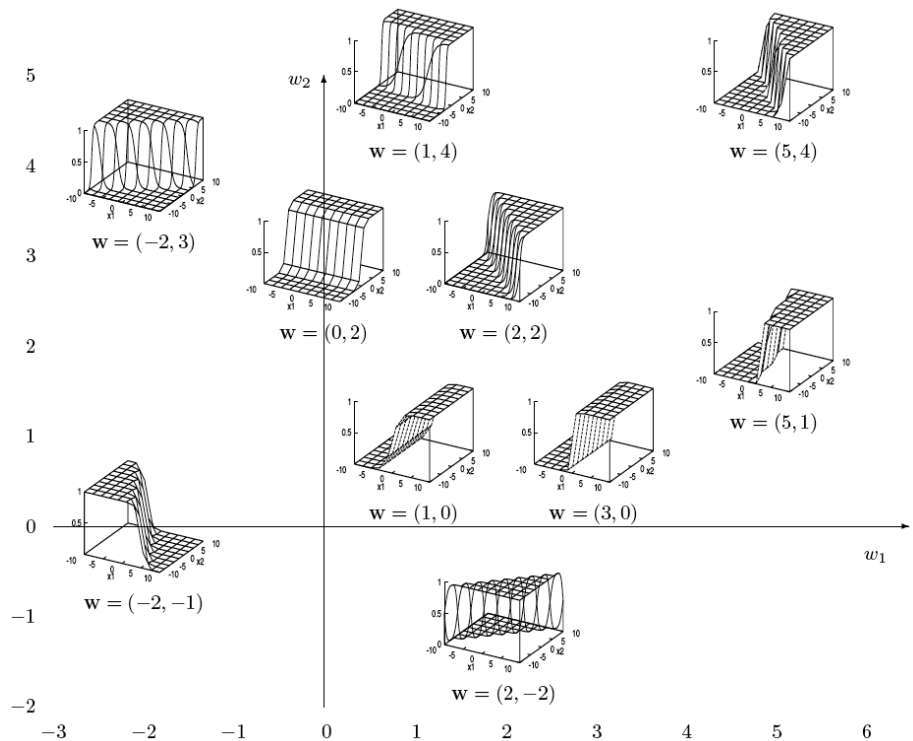
**Input space**

$$y(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2)}}$$



$\mathbf{w} = (0, 2)$

**Weight space**



}

# Single neuron as a classifier

## Concepts

### **The supervised learning paradigm:**

given example inputs  $\mathbf{x}$  and target outputs  $t$   
learning the mapping between them

the trained network is supposed to give  
'correct response' for *any given* input  
stimulus

training is equivalent to learning the  
appropriate weights

in order to achieve this, an *objective function* (or *error function*) is defined,  
which is minimized during training

# Binary classification in a neuron

**Error function:**

$$G(\mathbf{w}) = - \sum_n \left[ t^{(n)} \ln y(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

# Binary classification in a neuron

## Error function:

$$G(\mathbf{w}) = - \sum_n \left[ t^{(n)} \ln y(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

1. information content of outcomes
2. relative entropy of distributions  $(t, 1-t)$  and  $(y, 1-y)$

# Binary classification in a neuron

## Error function:

$$G(\mathbf{w}) = - \sum_n \left[ t^{(n)} \ln y(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

1. information content of outcomes
2. relative entropy of distributions  $(t, 1-t)$  and  $(y, 1-y)$

## Learning: back-propagation algorithm

$$g_j = \frac{\partial G}{\partial w_j} = \sum_{n=1}^N -(t^{(n)} - y^{(n)}) x_j^{(n)}$$

Here  $e^{(n)} \equiv t^{(n)} - y^{(n)}$  is the error



# Binary classification

## The algorithm

1. Get the data
2. Calculate activations:  $a = \sum_i w_i x_i$ ,
3. Calculate the output of the neuron:  $y(a) = \frac{1}{1 + e^{-a}}$
4. Calculate error:  $e = t - y$
5. Update weight(s):  $\Delta w_i = \eta e x_i$

# Binary classification

## The algorithm

1. Get the data
2. Calculate activations:  $a = \sum_i w_i x_i$ ,
3. Calculate the output of the neuron:  $y(a) = \frac{1}{1 + e^{-a}}$
4. Calculate error:  $e = t - y$
5. Update weight(s):  $\Delta w_i = \eta e x_i$

alternatively, batch learning can be done

$$g_i^{(n)} = -e^{(n)} x_i^{(n)}.$$
$$\Delta w_i = -\eta \sum_n g_i^{(n)}$$

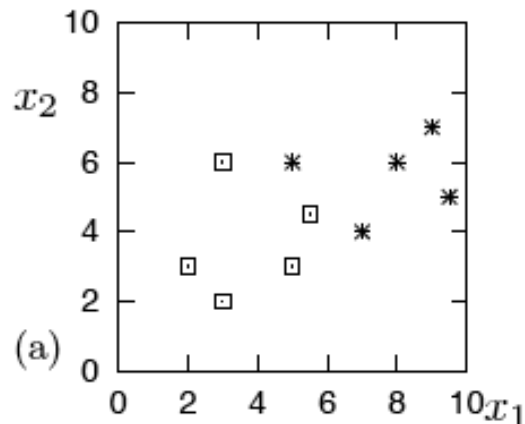
# Binary classification

## The algorithm

1. Get the data
2. Calculate activations:  $a = \sum_i w_i x_i$ ,
3. Calculate the output of the neuron:  $y(a) = \frac{1}{1 + e^{-a}}$
4. Calculate error:  $e = t - y$
5. Update weight(s):  $\Delta w_i = \eta e x_i$

alternatively, batch learning can be done  $g_i^{(n)} = -e^{(n)} x_i^{(n)}$ .

$$\Delta w_i = -\eta \sum_n g_i^{(n)}$$



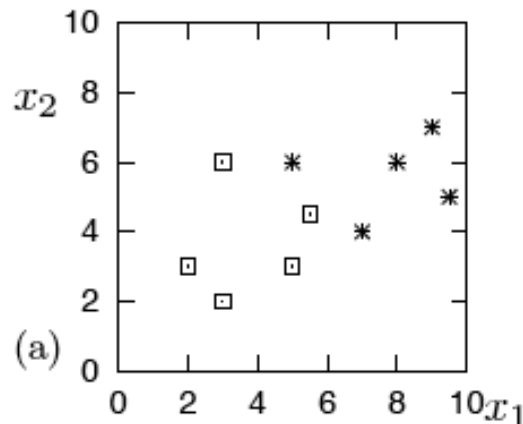
# Binary classification

## The algorithm

1. Get the data
2. Calculate activations:  $a = \sum_i w_i x_i$ ,
3. Calculate the output of the neuron:  $y(a) = \frac{1}{1 + e^{-a}}$
4. Calculate error:  $e = t - y$
5. Update weight(s):  $\Delta w_i = \eta e x_i$

alternatively, batch learning can be done  $g_i^{(n)} = -e^{(n)} x_i^{(n)}$ .

$$\Delta w_i = -\eta \sum_n g_i^{(n)}$$



gradient descent

# Binary classification

## The algorithm

1. Get the data

2. Calculate activations:  $a = \sum_i w_i x_i$ ,

3. Calculate the output of the neuron:  $y(a) = \frac{1}{1 + e^{-a}}$

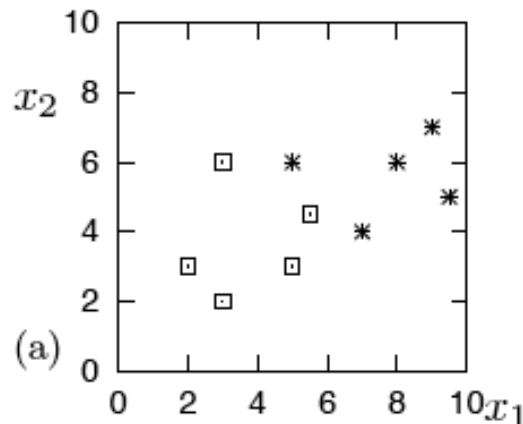
4. Calculate error:  $e = t - y$

5. Update weight(s):  $\Delta w_i = \eta e x_i$

stochastic gradient descent

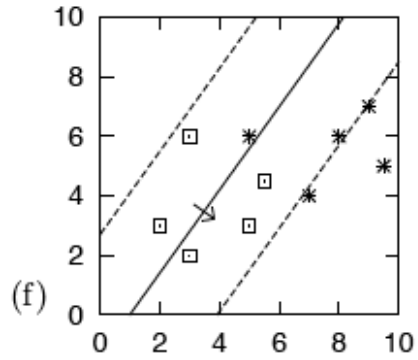
alternatively, batch learning can be done  $g_i^{(n)} = -e^{(n)} x_i^{(n)}$ .

$$\Delta w_i = -\eta \sum_n g_i^{(n)}$$

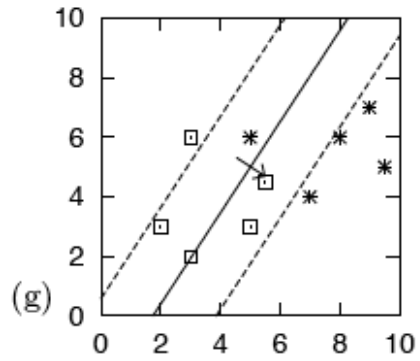
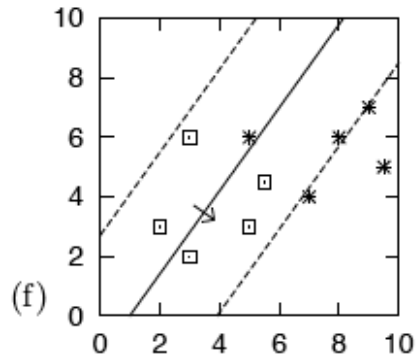


gradient descent

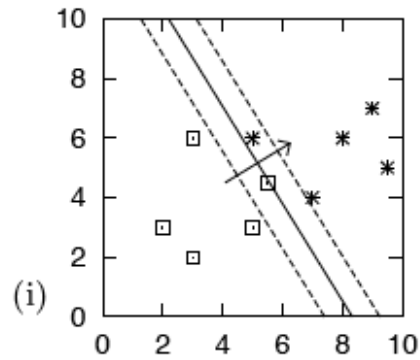
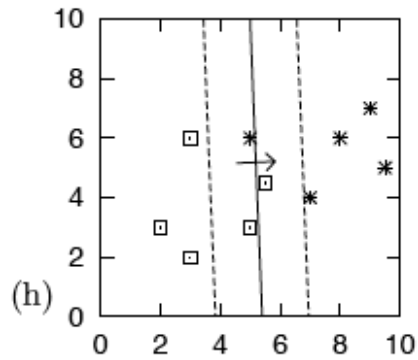
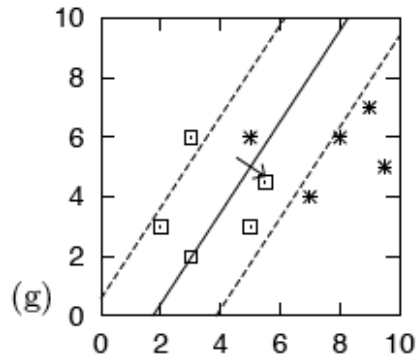
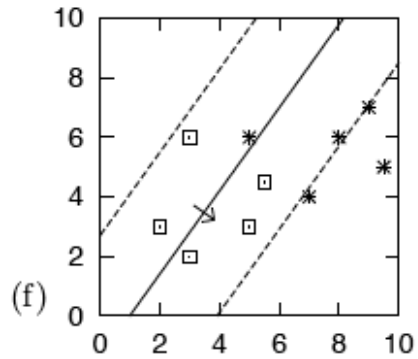
# Binary classification Demonstration



# Binary classification Demonstration

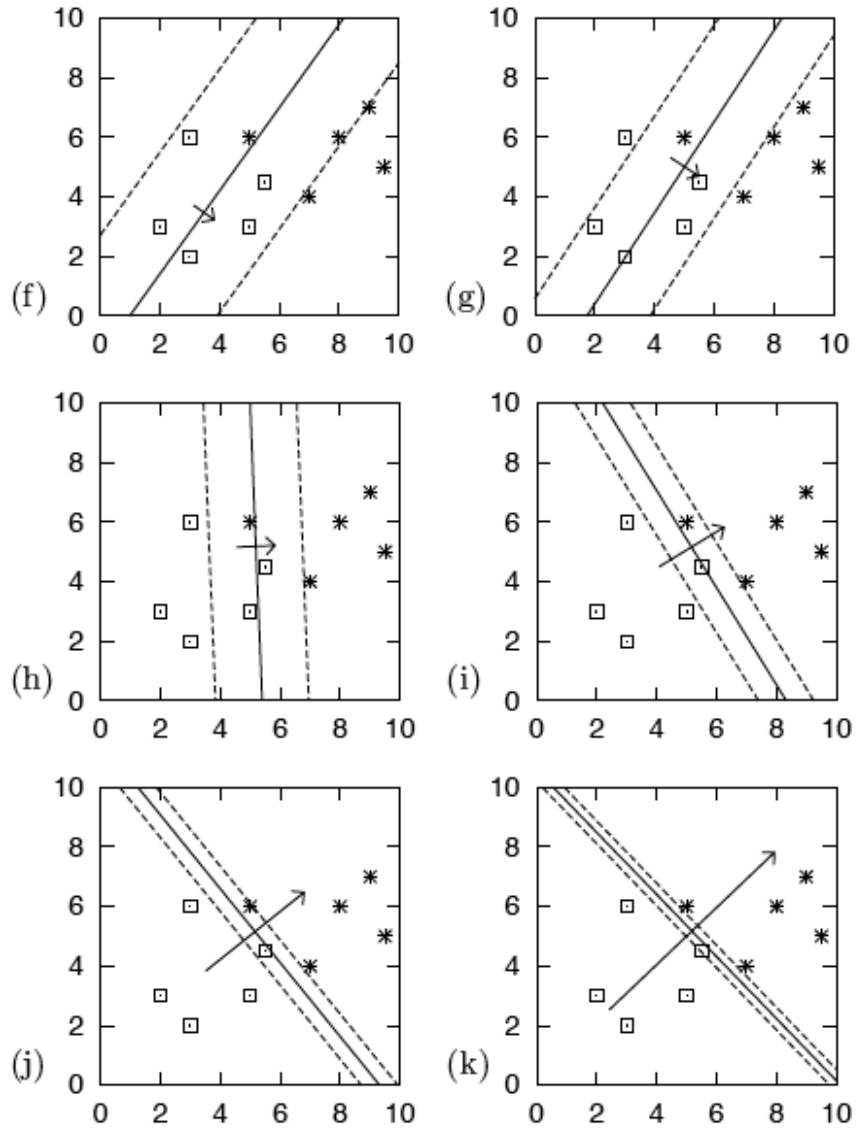


# Binary classification Demonstration

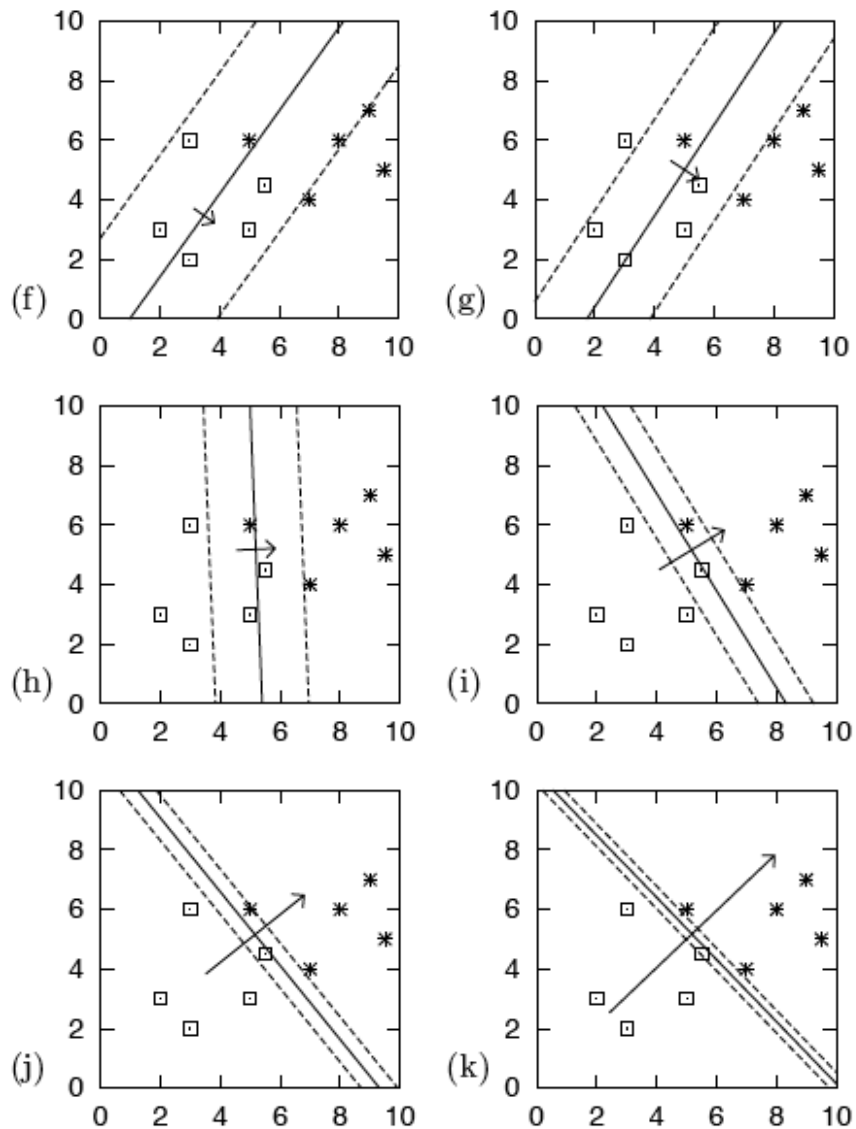




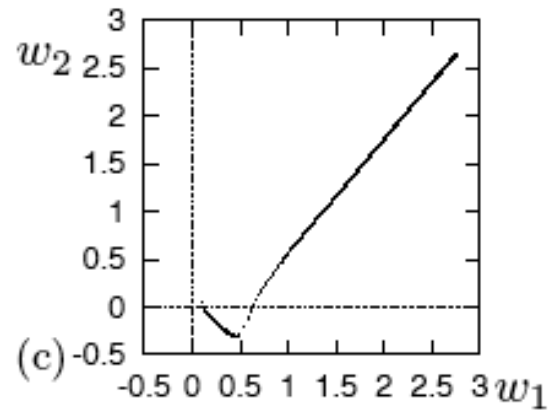
# Binary classification Demonstration



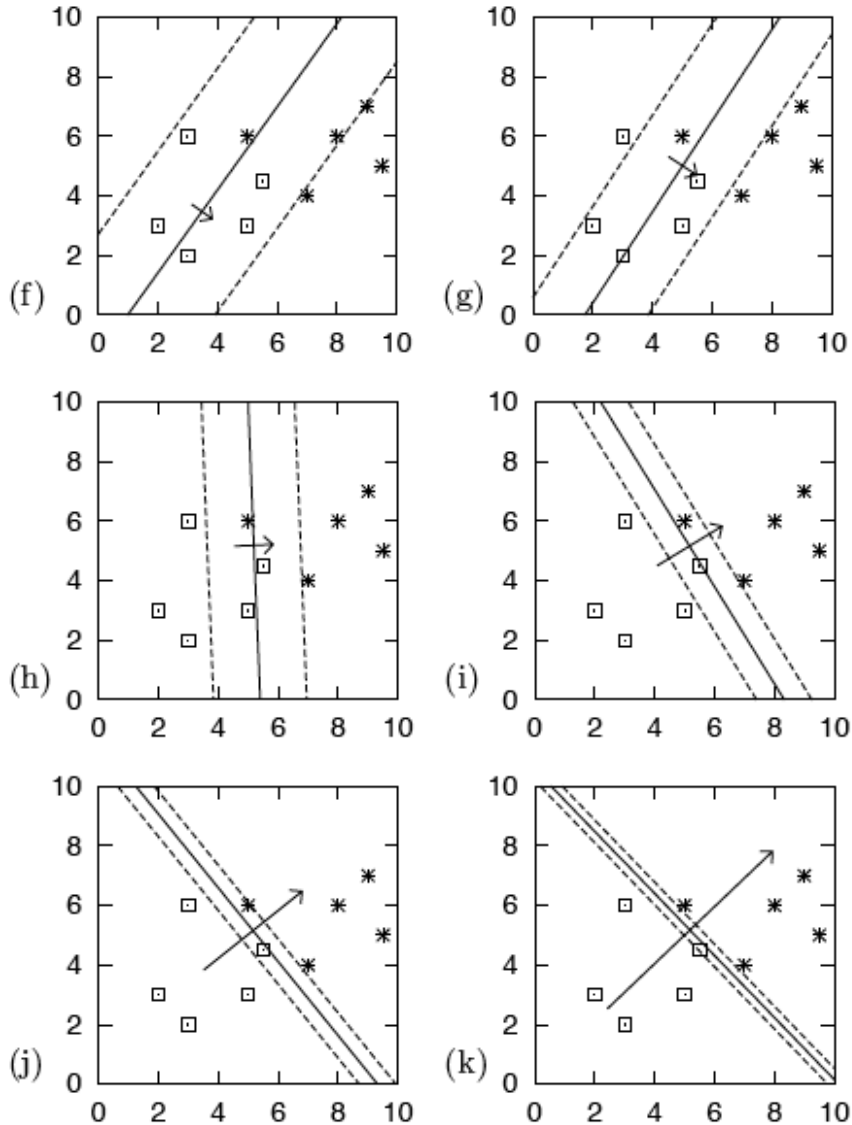
# Binary classification Demonstration



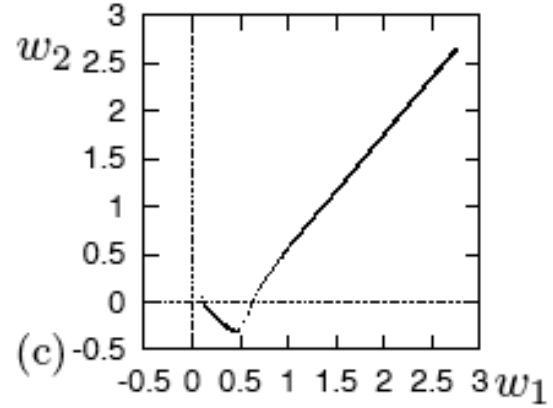
Evolution of weights



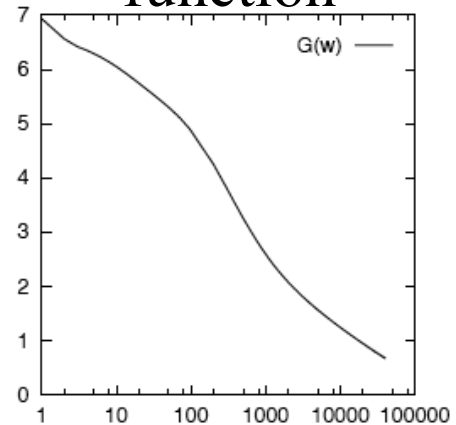
# Binary classification Demonstration



## Evolution of weights

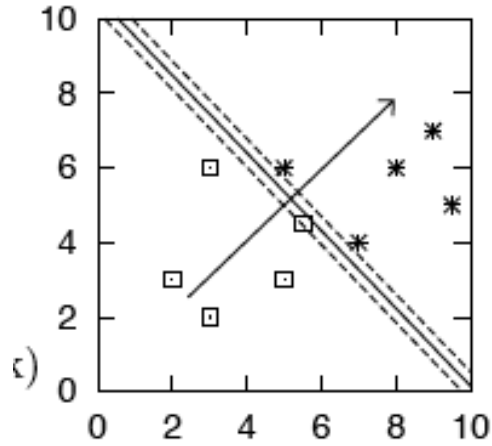


## Evolution of the error function



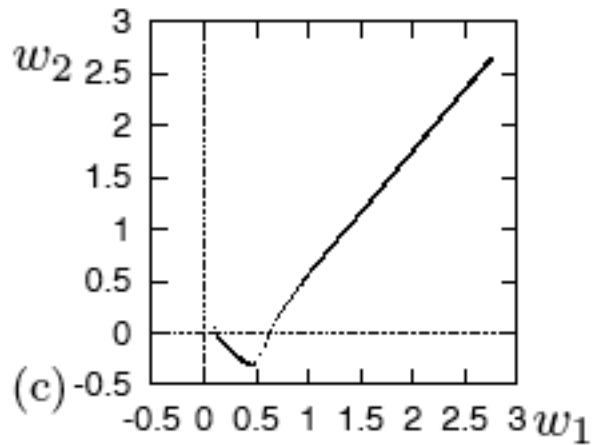
# Binary classification

## Are we happy?



# Binary classification

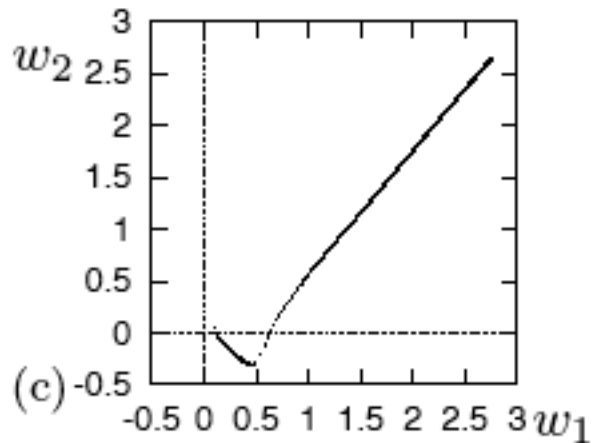
## Are we happy?



$w$ 's seem to grow unbounded

# Binary classification

## Are we happy?

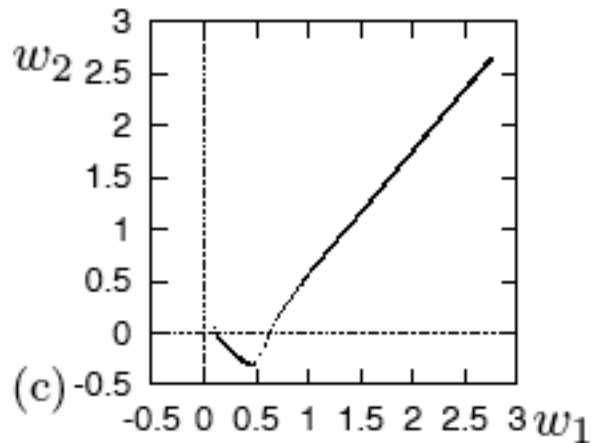


$w$ 's seem to grow unbounded

Is this bad for us?

# Binary classification

## Are we happy?



$w$ 's seem to grow unbounded

Is this bad for us?

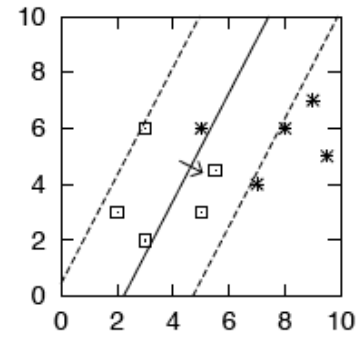
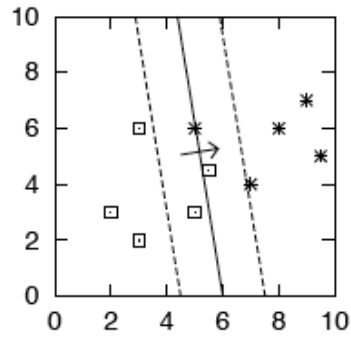
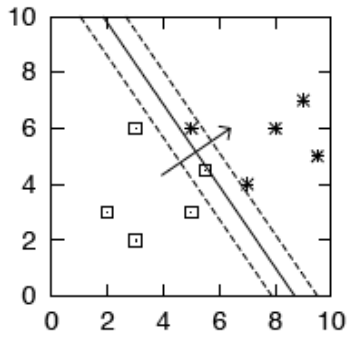
Generic pain killer: *regularization*

$$M(\mathbf{w}) = G(\mathbf{w}) + \alpha E_W(\mathbf{w})$$

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_i w_i^2. \quad (\text{weight decay})$$

# Binary classification

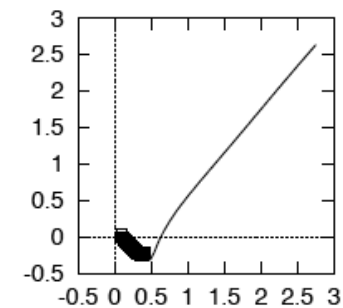
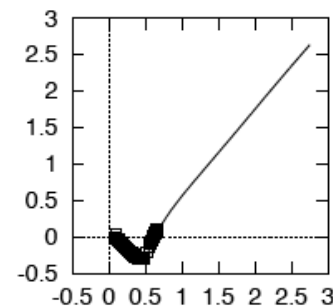
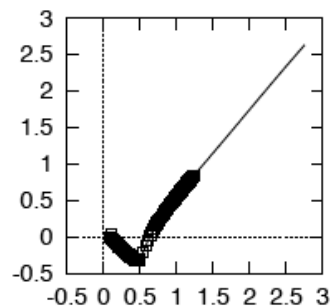
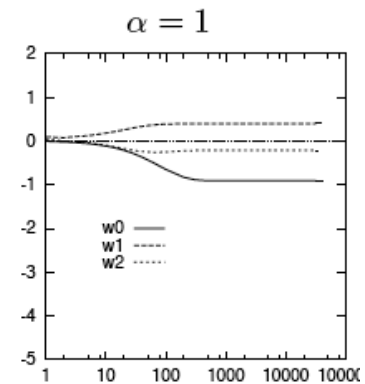
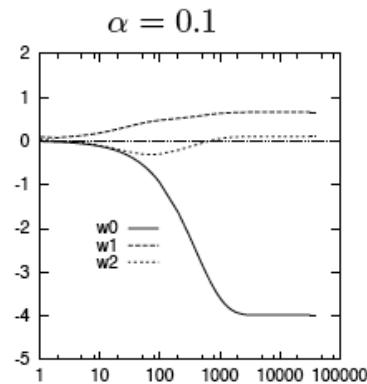
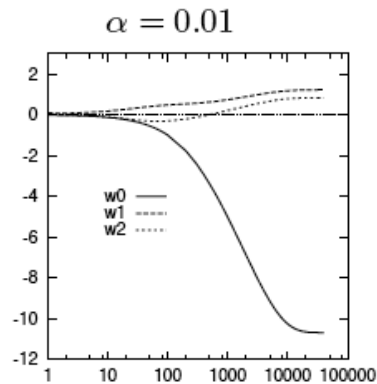
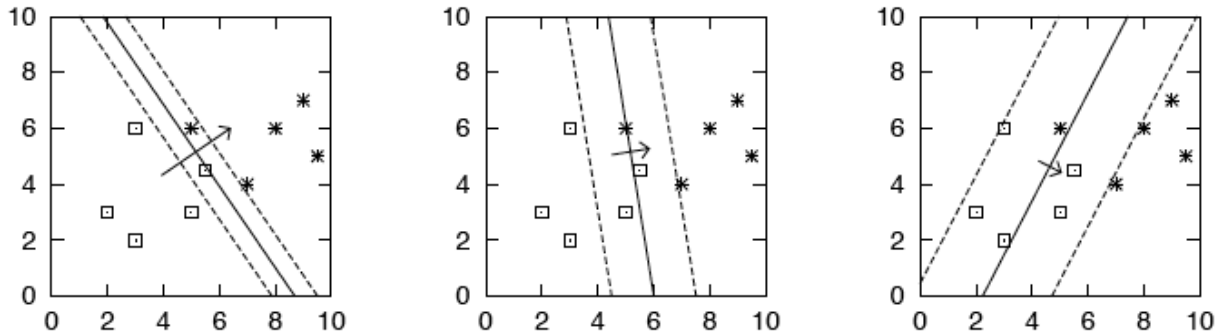
## Effect of regularization



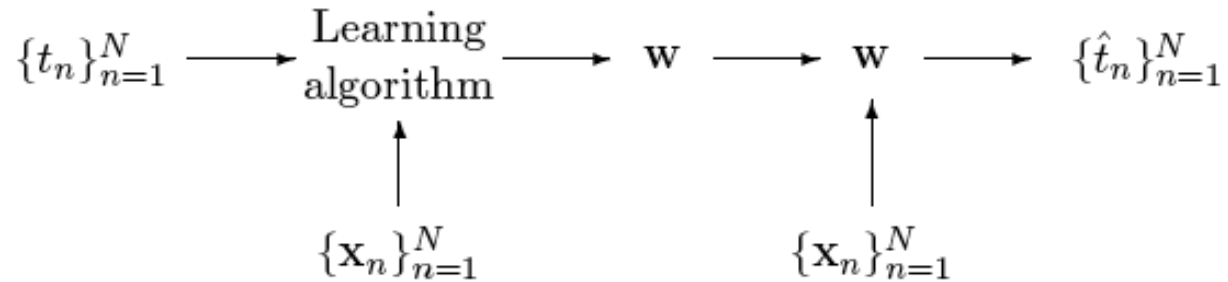


# Binary classification

## Effect of regularization



# Capacity of the neuron



How much information can the neuron transmit?

# Interpreting learning as inference

So far: optimization wrt. an objective function

$$M(\mathbf{w}) = G(\mathbf{w}) + \alpha E_W(\mathbf{w})$$

where

$$G(\mathbf{w}) = - \sum_n \left[ t^{(n)} \ln y(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_i w_i^2.$$

# Interpreting learning as inference

So far: optimization wrt. an objective function

$$M(\mathbf{w}) = G(\mathbf{w}) + \alpha E_W(\mathbf{w})$$

where

$$G(\mathbf{w}) = - \sum_n \left[ t^{(n)} \ln y(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_i w_i^2.$$

**What's this quirky regularizer, anyway?**

# Interpreting learning as inference

Let's interpret  $y(\mathbf{x}, \mathbf{w})$  as a probability:

$$\begin{aligned}P(t = 1 \mid \mathbf{w}, \mathbf{x}) &= y \\P(t = 0 \mid \mathbf{w}, \mathbf{x}) &= 1 - y\end{aligned}$$

# Interpreting learning as inference

Let's interpret  $y(\mathbf{x}, \mathbf{w})$  as a probability:

$$\begin{aligned}P(t = 1 | \mathbf{w}, \mathbf{x}) &= y \\P(t = 0 | \mathbf{w}, \mathbf{x}) &= 1 - y\end{aligned}$$

in a compact form:

$$P(t | \mathbf{w}, \mathbf{x}) = y^t (1 - y)^{1-t} = \exp[t \ln y + (1 - t) \ln(1 - y)]$$

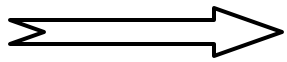
# Interpreting learning as inference

Let's interpret  $y(\mathbf{x}, \mathbf{w})$  as a probability:

$$\begin{aligned}P(t = 1 | \mathbf{w}, \mathbf{x}) &= y \\P(t = 0 | \mathbf{w}, \mathbf{x}) &= 1 - y\end{aligned}$$

in a compact form:

$$P(t | \mathbf{w}, \mathbf{x}) = y^t (1 - y)^{1-t} = \exp[t \ln y + (1 - t) \ln(1 - y)]$$



the **likelihood** of the input data can be expressed with the original error function function

$$P(D | \mathbf{w}) = \exp[-G(\mathbf{w})]$$

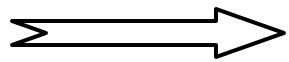
# Interpreting learning as inference

Let's interpret  $y(\mathbf{x}, \mathbf{w})$  as a probability:

$$\begin{aligned}P(t = 1 | \mathbf{w}, \mathbf{x}) &= y \\P(t = 0 | \mathbf{w}, \mathbf{x}) &= 1 - y\end{aligned}$$

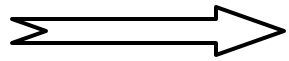
in a compact form:

$$P(t | \mathbf{w}, \mathbf{x}) = y^t (1 - y)^{1-t} = \exp[t \ln y + (1 - t) \ln(1 - y)]$$



the **likelihood** of the input data can be expressed with the original error function

$$P(D | \mathbf{w}) = \exp[-G(\mathbf{w})]$$



the regularizer has the form of a prior!

$$P(\mathbf{w} | \alpha) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W).$$



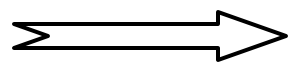
# Interpreting learning as inference

Let's interpret  $y(\mathbf{x}, \mathbf{w})$  as a probability:

$$\begin{aligned}P(t = 1 | \mathbf{w}, \mathbf{x}) &= y \\P(t = 0 | \mathbf{w}, \mathbf{x}) &= 1 - y\end{aligned}$$

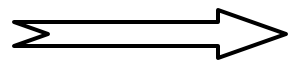
in a compact form:

$$P(t | \mathbf{w}, \mathbf{x}) = y^t (1 - y)^{1-t} = \exp[t \ln y + (1 - t) \ln(1 - y)]$$



the **likelihood** of the input data can be expressed with the original error function

$$P(D | \mathbf{w}) = \exp[-G(\mathbf{w})]$$



the regularizer has the form of a prior!

$$P(\mathbf{w} | \alpha) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W).$$

what we get in the objective function  $M(\mathbf{w})$ :

the posterior distribution of  $\mathbf{w}$ : 
$$P(\mathbf{w} | D, \alpha) = \frac{P(D | \mathbf{w})P(\mathbf{w} | \alpha)}{P(D | \alpha)}$$

# Interpreting learning as inference

## Relationship between $M(\mathbf{w})$ and the posterior

$$\begin{aligned} P(\mathbf{w} | D, \alpha) &= \frac{P(D | \mathbf{w})P(\mathbf{w} | \alpha)}{P(D | \alpha)} \\ &= \frac{e^{-G(\mathbf{w})} e^{-\alpha E_W(\mathbf{w})} / Z_W(\alpha)}{P(D | \alpha)} \\ &= \frac{1}{Z_M} \exp(-M(\mathbf{w})). \end{aligned}$$

**interpretation:** minimizing  $M(\mathbf{w})$  leads to finding the *maximum a posteriori* estimate  $\mathbf{w}_{\text{MP}}$

The log probability interpretation of the objective function retains:  
additivity of errors, while  
keeping the multiplicativity of probabilities

# Interpreting learning as inference

## Properties of the Bayesian estimate

# Interpreting learning as inference

## Properties of the Bayesian estimate

- The probabilistic interpretation makes our assumptions explicit:  
by the regularizer we imposed a soft constraint on the learned parameters, which expresses our *prior expectations*.
- An additional plus:  
beyond getting  $\mathbf{w}_{\text{MP}}$  we get a measure for learned parameter uncertainty