

# Machine Learning

Prof. Barbara Caputo

Dip. Ingegneria Informatica, Automatica e Gestionale, Roma

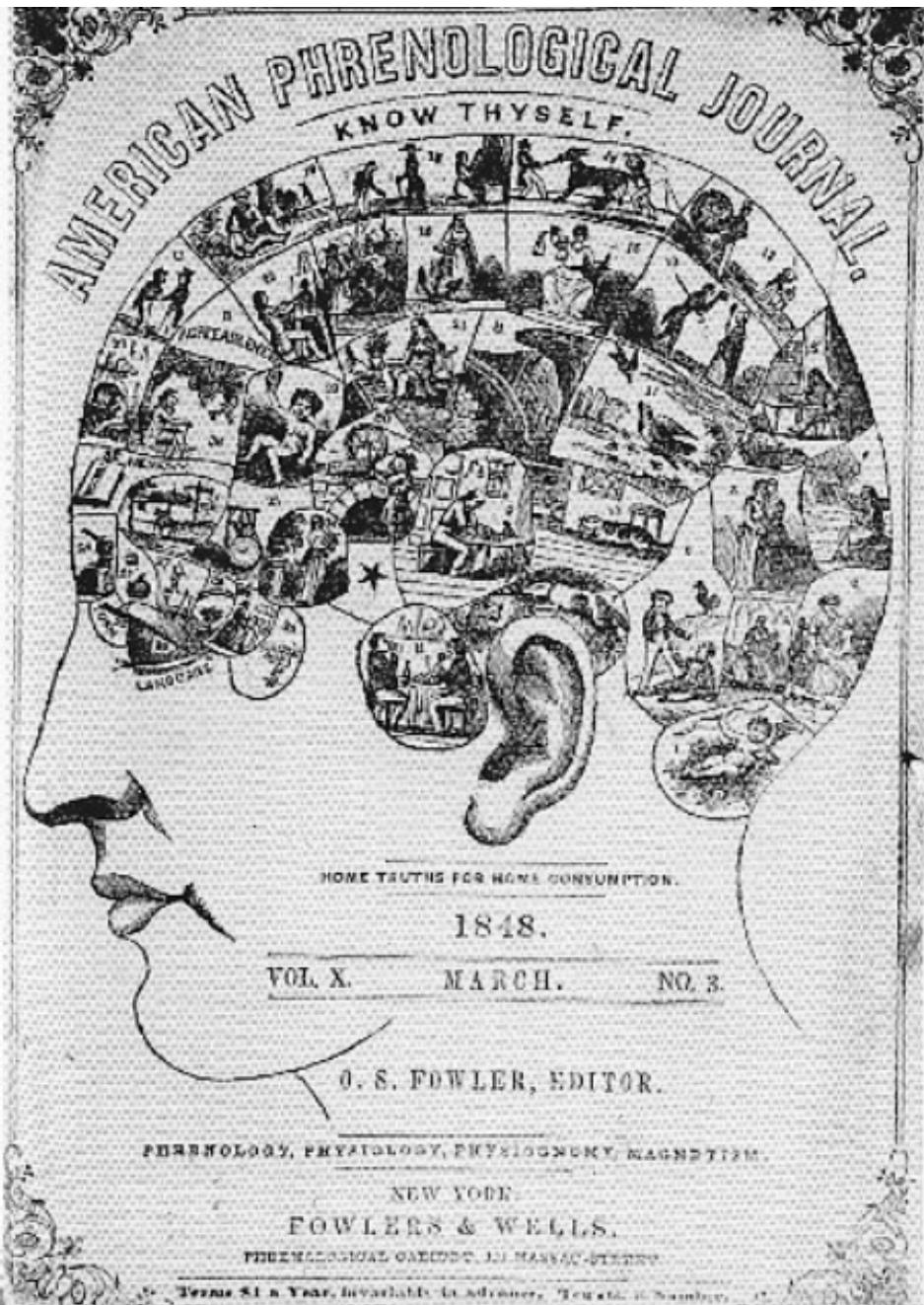


SAPIENZA  
UNIVERSITÀ DI ROMA

# Outline

- Perceptron
  - Hebbian learning & biology
  - Algorithm
  - Convergence analysis
- Features and preprocessing
  - Nonlinear separation
  - Perceptron in feature space
- Kernels
  - Kernel trick
  - Properties
  - Examples

# Perceptron



## early theories of the brain

# Biology and Learning

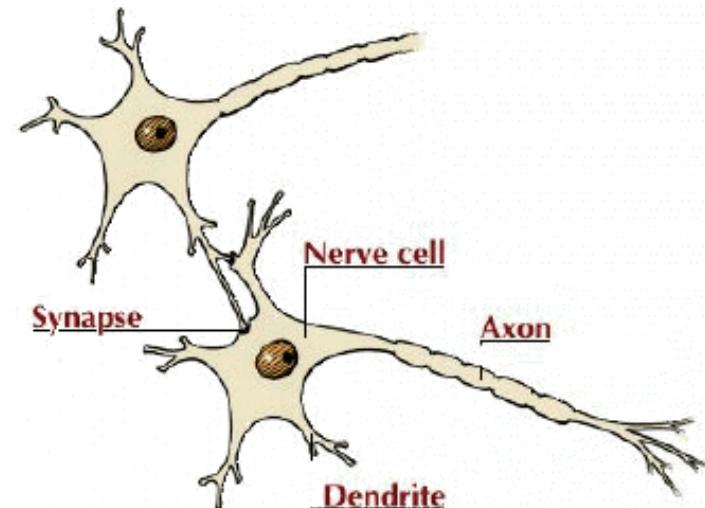
- Basic Idea
  - Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves system fitness.
  - Killing a sabertooth tiger should be rewarded ...
  - Correlated events should be combined.
  - Pavlov's salivating dog.

# Biology and Learning

- Basic Idea
  - Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves system fitness.
  - Killing a sabertooth tiger should be rewarded ...
  - Correlated events should be combined.
  - Pavlov's salivating dog.
- Training mechanisms
  - Behavioral modification of individuals (learning)  
Successful behavior is rewarded (e.g. food).
  - Hard-coded behavior in the genes (instinct)  
The wrongly coded animal does not reproduce.

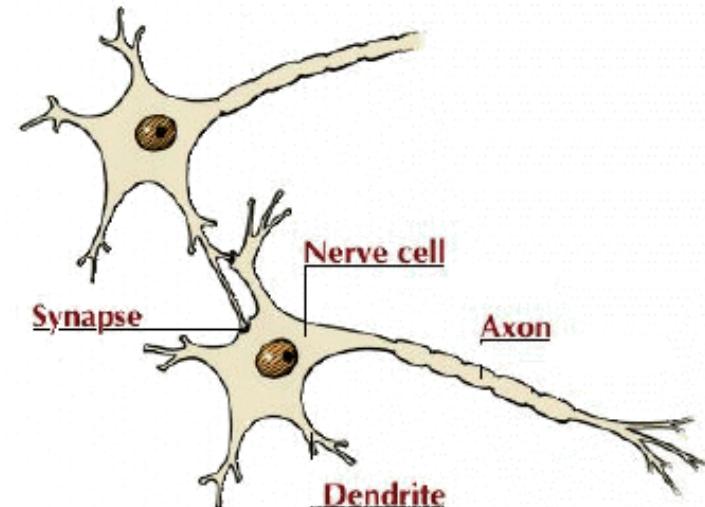
# Neurons

- Soma (CPU)  
Cell body - combines signals



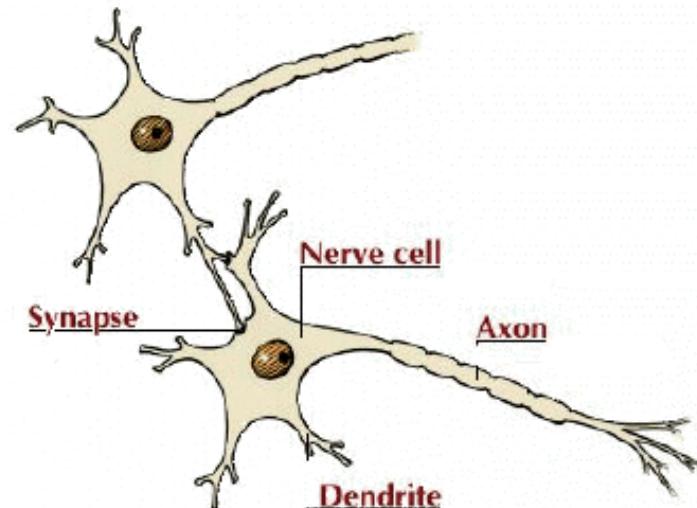
# Neurons

- Soma (CPU)  
Cell body - combines signals
- Dendrite (input bus)  
Combines the inputs from several other nerve cells



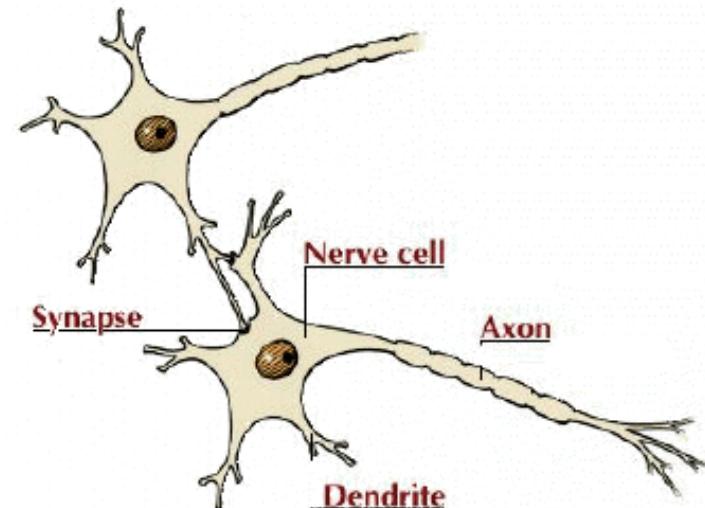
# Neurons

- Soma (CPU)  
Cell body - combines signals
- Dendrite (input bus)  
Combines the inputs from several other nerve cells
- Synapse (interface)  
Interface and **parameter store between neurons**

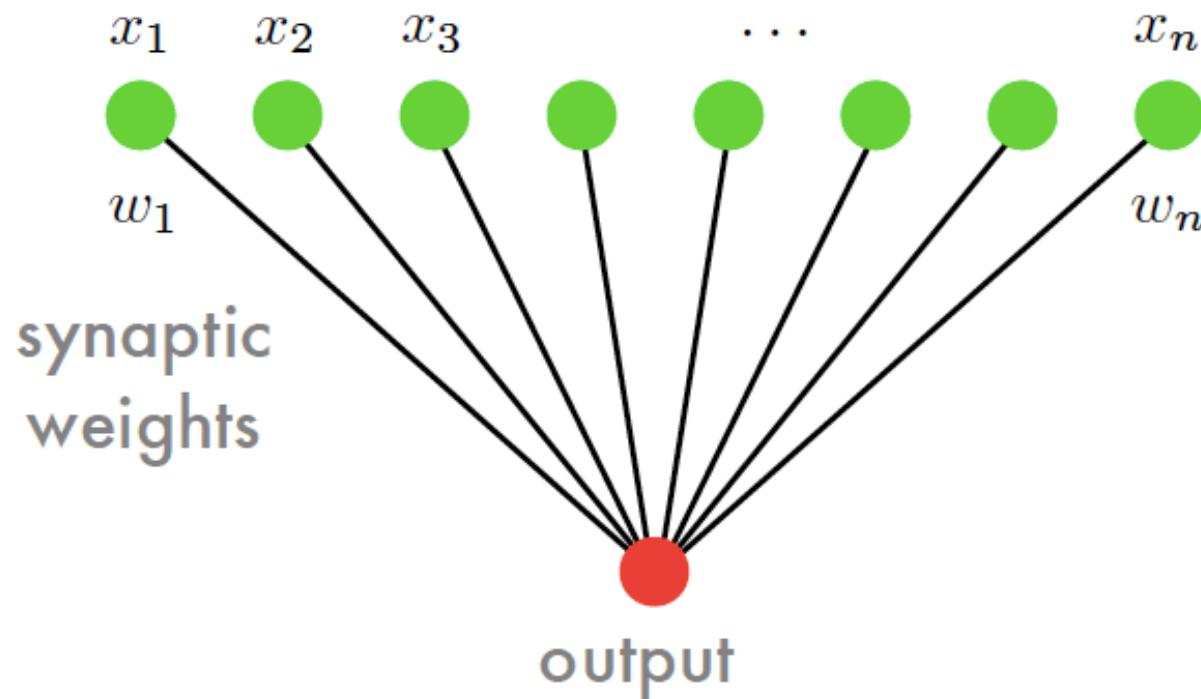


# Neurons

- Soma (CPU)  
Cell body - combines signals
- Dendrite (input bus)  
Combines the inputs from several other nerve cells
- Synapse (interface)  
Interface and **parameter store** between neurons
- Axon (cable)  
May be up to 1m long and will transport the activation signal to neurons at different locations



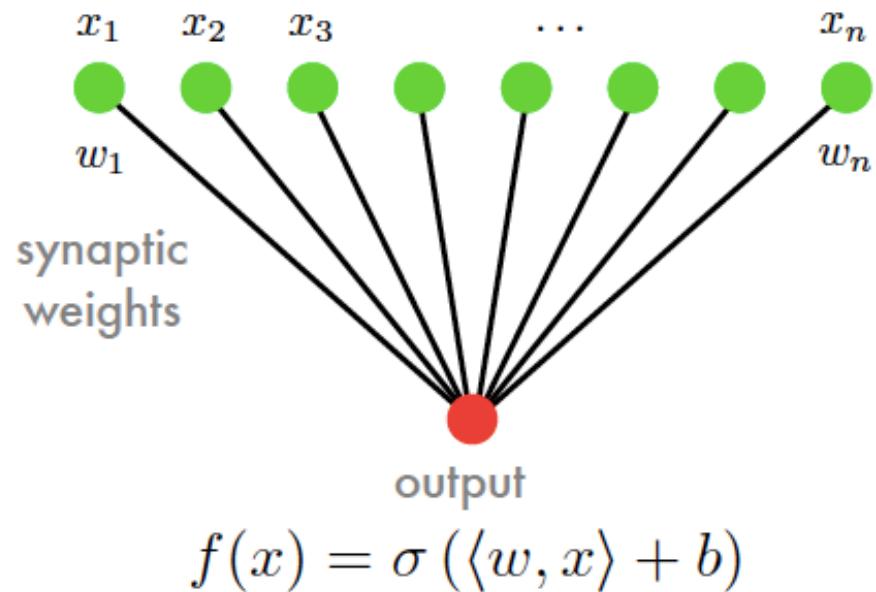
# Neurons



$$f(x) = \sum_i w_i x_i = \langle w, x \rangle$$

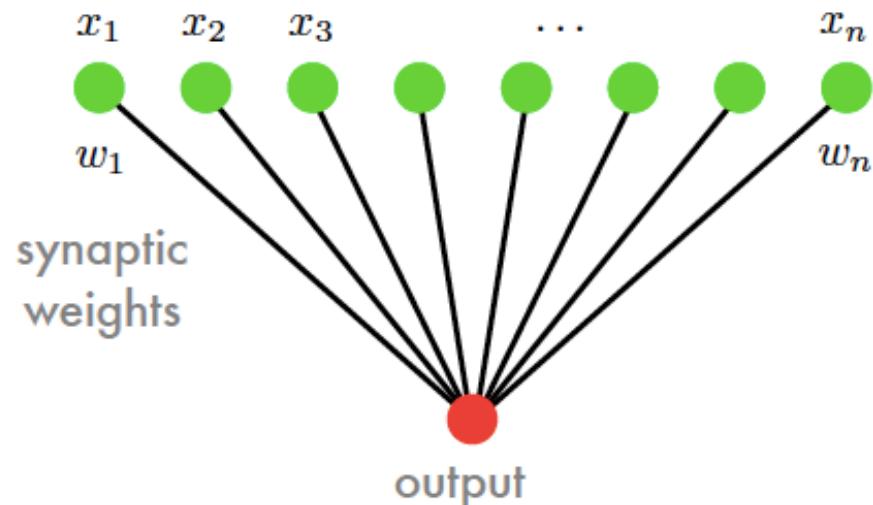
# Perceptron

- Weighted linear combination



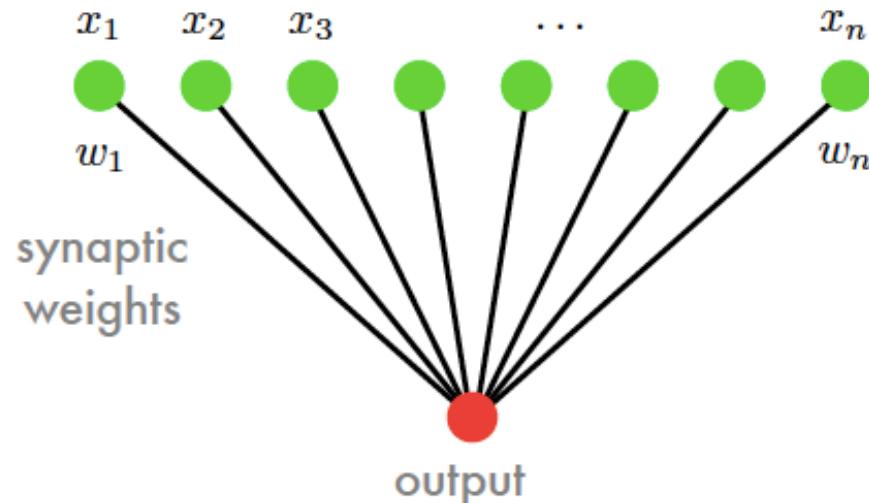
# Perceptron

- Weighted linear combination
- Nonlinear decision function



# Perceptron

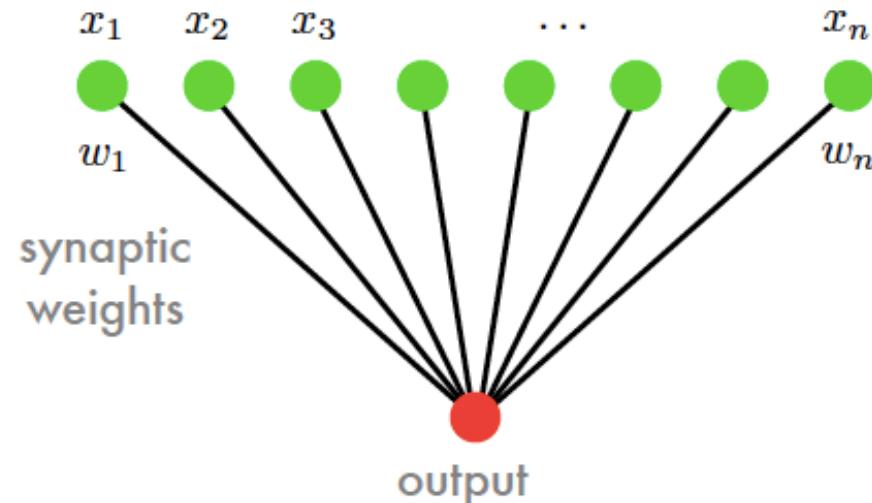
- Weighted linear combination
- Nonlinear decision function
- Linear offset (bias)



- Linear separating hyperplanes (spam/ham, novel/typical, click/no click)

# Perceptron

- Weighted linear combination
- Nonlinear decision function
- Linear offset (bias)

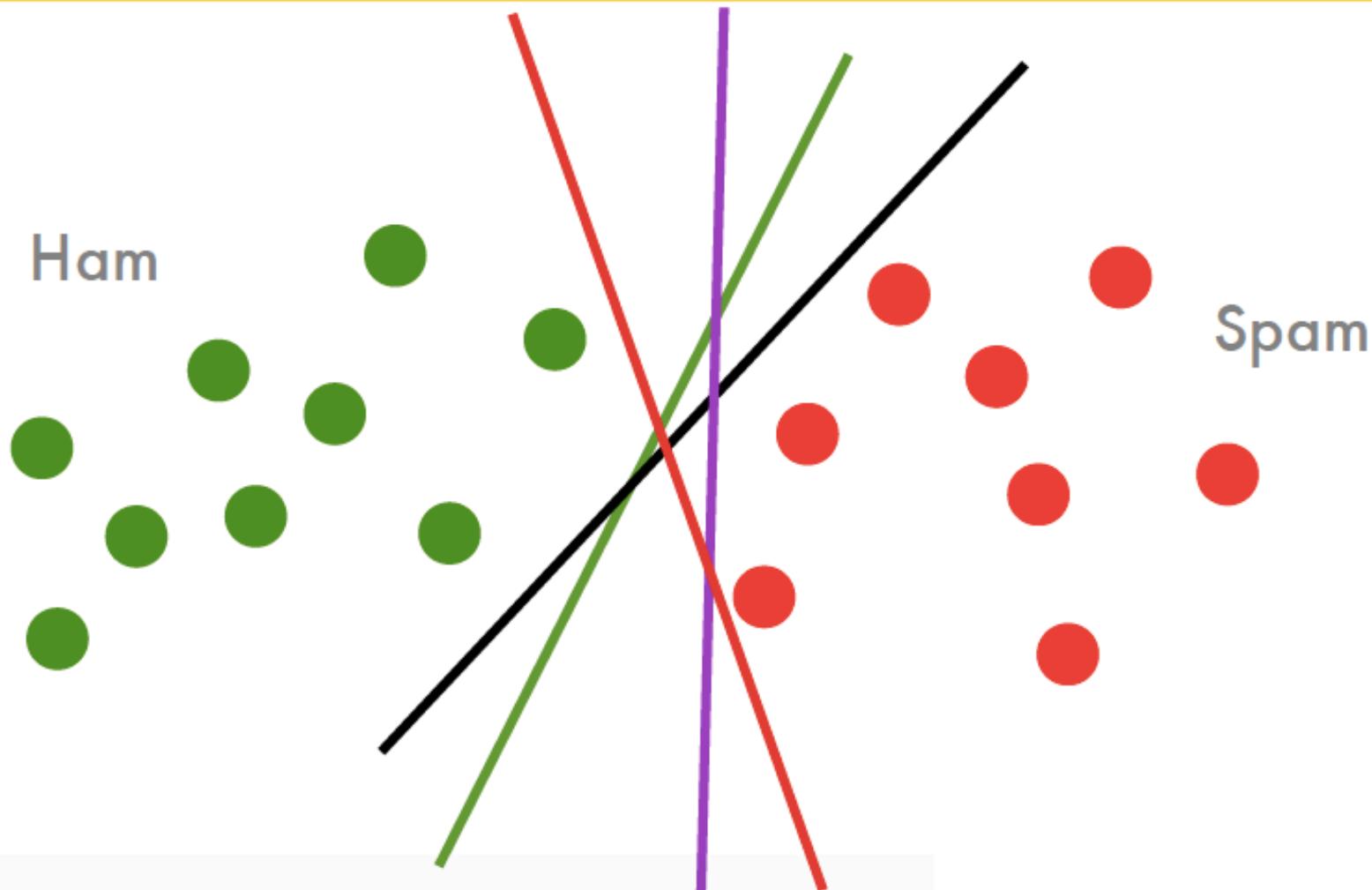


$$f(x) = \sigma(\langle w, x \rangle + b)$$

- Linear separating hyperplanes (spam/ham, novel/typical, click/no click)
- Learning

Estimating the parameters  $w$  and  $b$

# Perceptron



# The Perceptron

**initialize**  $w = 0$  and  $b = 0$

**repeat**

**if**  $y_i [\langle w, x_i \rangle + b] \leq 0$  **then**

$w \leftarrow w + y_i x_i$  and  $b \leftarrow b + y_i$

**end if**

**until** all classified correctly

# The Perceptron

**initialize**  $w = 0$  and  $b = 0$

**repeat**

**if**  $y_i [\langle w, x_i \rangle + b] \leq 0$  **then**

$w \leftarrow w + y_i x_i$  and  $b \leftarrow b + y_i$

**end if**

**until** all classified correctly

- **Nothing happens if classified correctly**

# The Perceptron

initialize  $w = 0$  and  $b = 0$

repeat

if  $y_i [\langle w, x_i \rangle + b] \leq 0$  then  
     $w \leftarrow w + y_i x_i$  and  $b \leftarrow b + y_i$   
end if

until all classified correctly

- Nothing happens if classified correctly
- Weight vector is linear combination  $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products  $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$

# Convergence Theorem

- If there exists some  $(w^*, b^*)$  with unit length and

$$y_i [\langle x_i, w^* \rangle + b^*] \geq \rho \text{ for all } i$$

# Convergence Theorem

- If there exists some  $(w^*, b^*)$  with unit length and  
 $y_i [\langle x_i, w^* \rangle + b^*] \geq \rho$  for all  $i$   
then the perceptron converges to a linear separator after a number of steps bounded by

$$\left(b^{*2} + 1\right) (r^2 + 1) \rho^{-2} \text{ where } \|x_i\| \leq r$$

# Convergence Theorem

- If there exists some  $(w^*, b^*)$  with unit length and  
 $y_i [\langle x_i, w^* \rangle + b^*] \geq \rho$  for all  $i$   
then the perceptron converges to a linear separator after a number of steps bounded by
$$(b^{*2} + 1) (r^2 + 1) \rho^{-2}$$
 where  $\|x_i\| \leq r$
- Dimensionality independent

# Convergence Theorem

- If there exists some  $(w^*, b^*)$  with unit length and

$$y_i [\langle x_i, w^* \rangle + b^*] \geq \rho \text{ for all } i$$

then the perceptron converges to a linear separator after a number of steps bounded by

$$(b^{*2} + 1) (r^2 + 1) \rho^{-2} \text{ where } \|x_i\| \leq r$$

- Dimensionality independent
- Order independent (i.e. also worst case)
- Scales with 'difficulty' of problem

# Proof

## Starting Point

We start from  $w_1 = 0$  and  $b_1 = 0$ .

## Step 1: Bound on the increase of alignment

Denote by  $w_i$  the value of  $w$  at step  $i$  (analogously  $b_i$ ).

Alignment:  $\langle (w_i, b_i), (w^*, b^*) \rangle$

For error in observation  $(x_i, y_i)$  we get

$$\begin{aligned} & \langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle \\ &= \langle [(w_j, b_j) + y_i(x_i, 1)], (w^*, b^*) \rangle \\ &= \langle (w_j, b_j), (w^*, b^*) \rangle + y_i \langle (x_i, 1) \cdot (w^*, b^*) \rangle \\ &\geq \langle (w_j, b_j), (w^*, b^*) \rangle + \rho \\ &\geq j\rho. \end{aligned}$$

Alignment increases with number of errors.

# Proof

## Step 2: Cauchy-Schwartz for the Dot Product

$$\begin{aligned}\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle &\leq \|(w_{j+1}, b_{j+1})\| \|(w^*, b^*)\| \\ &= \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\|\end{aligned}$$

## Step 3: Upper Bound on $\|(w_j, b_j)\|$

If we make a mistake we have

$$\begin{aligned}\|(w_{j+1}, b_{j+1})\|^2 &= \|(w_j, b_j) + y_i(x_i, 1)\|^2 \\ &= \|(w_j, b_j)\|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \|(x_i, 1)\|^2 \\ &\leq \|(w_j, b_j)\|^2 + \|(x_i, 1)\|^2 \\ &\leq j(R^2 + 1).\end{aligned}$$

## Step 4: Combination of first three steps

$$j\rho \leq \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\| \leq \sqrt{j(R^2 + 1)((b^*)^2 + 1)}$$

Solving for  $j$  proves the theorem.

# Consequences

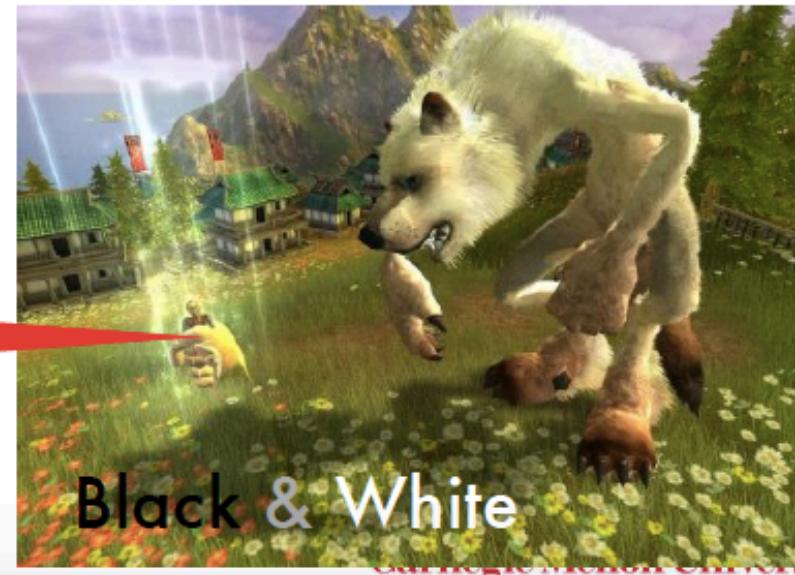
- Only need to store errors.  
This gives a compression bound for perceptron.

# Consequences

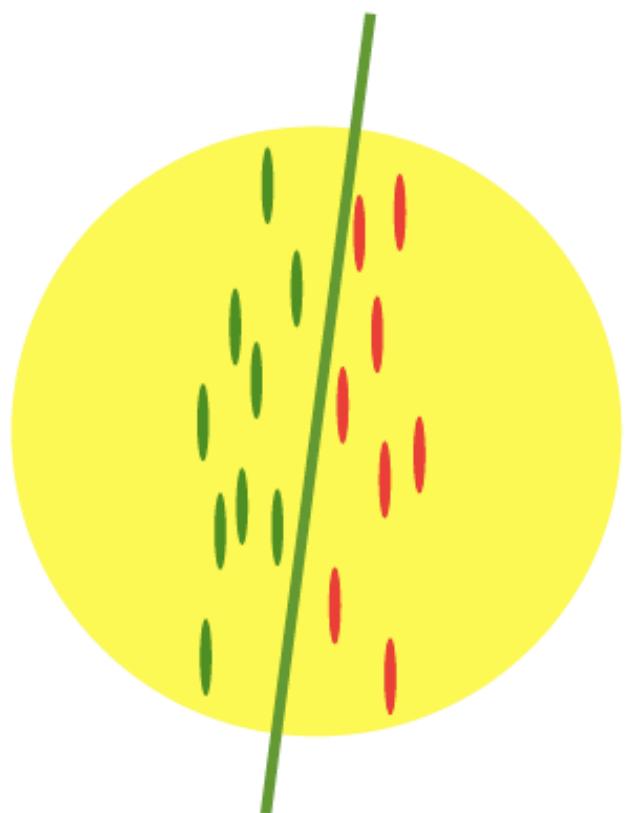
- Only need to store errors.  
This gives a compression bound for perceptron.

- Fails with noisy data

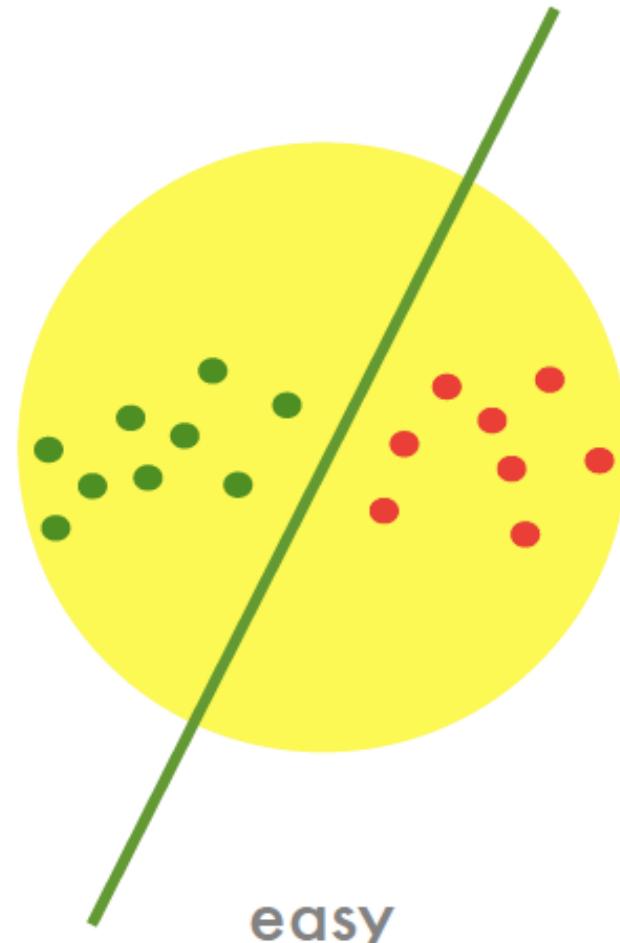
do NOT train your  
avatar with perceptrons



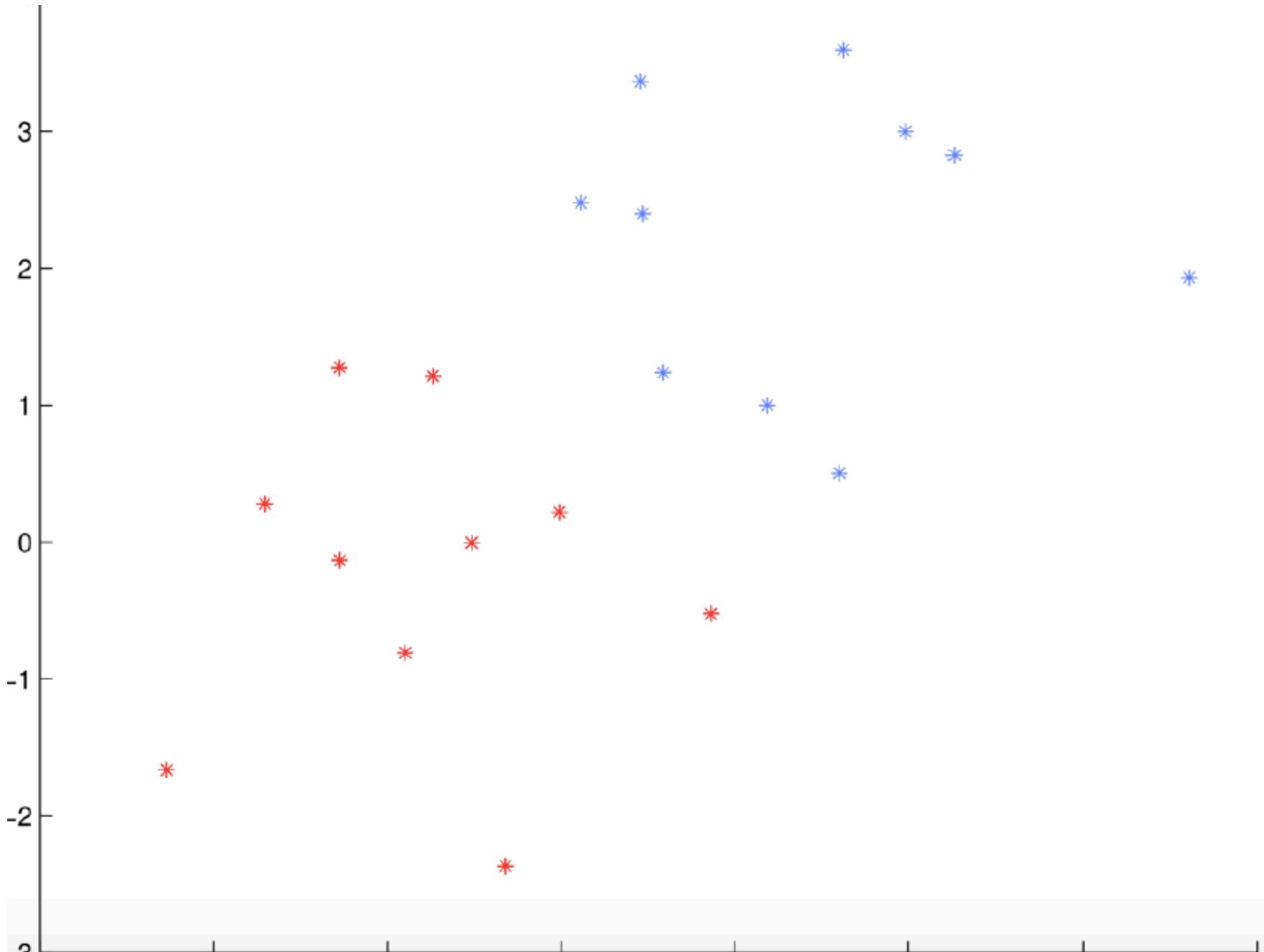
# Hardness margin vs. size

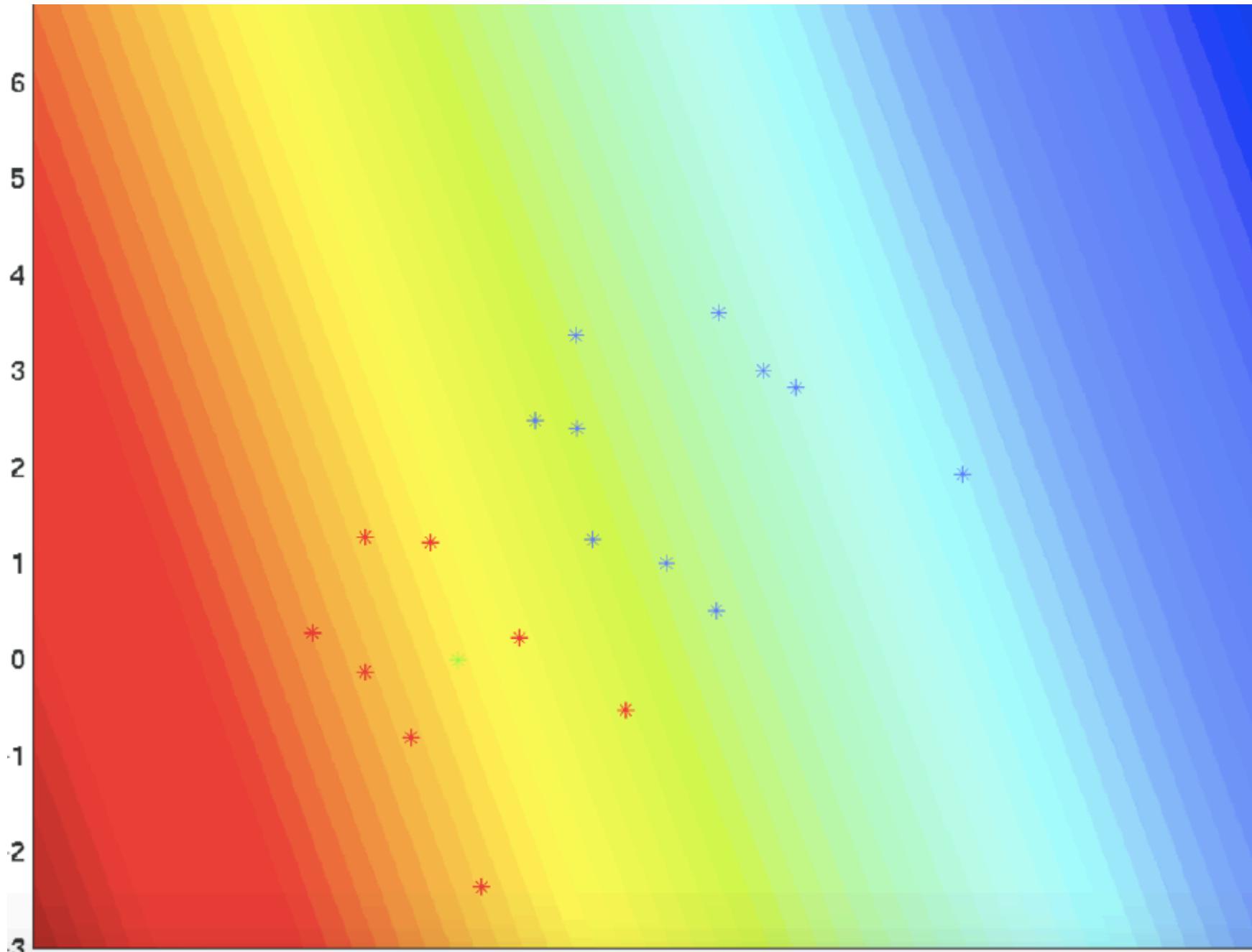


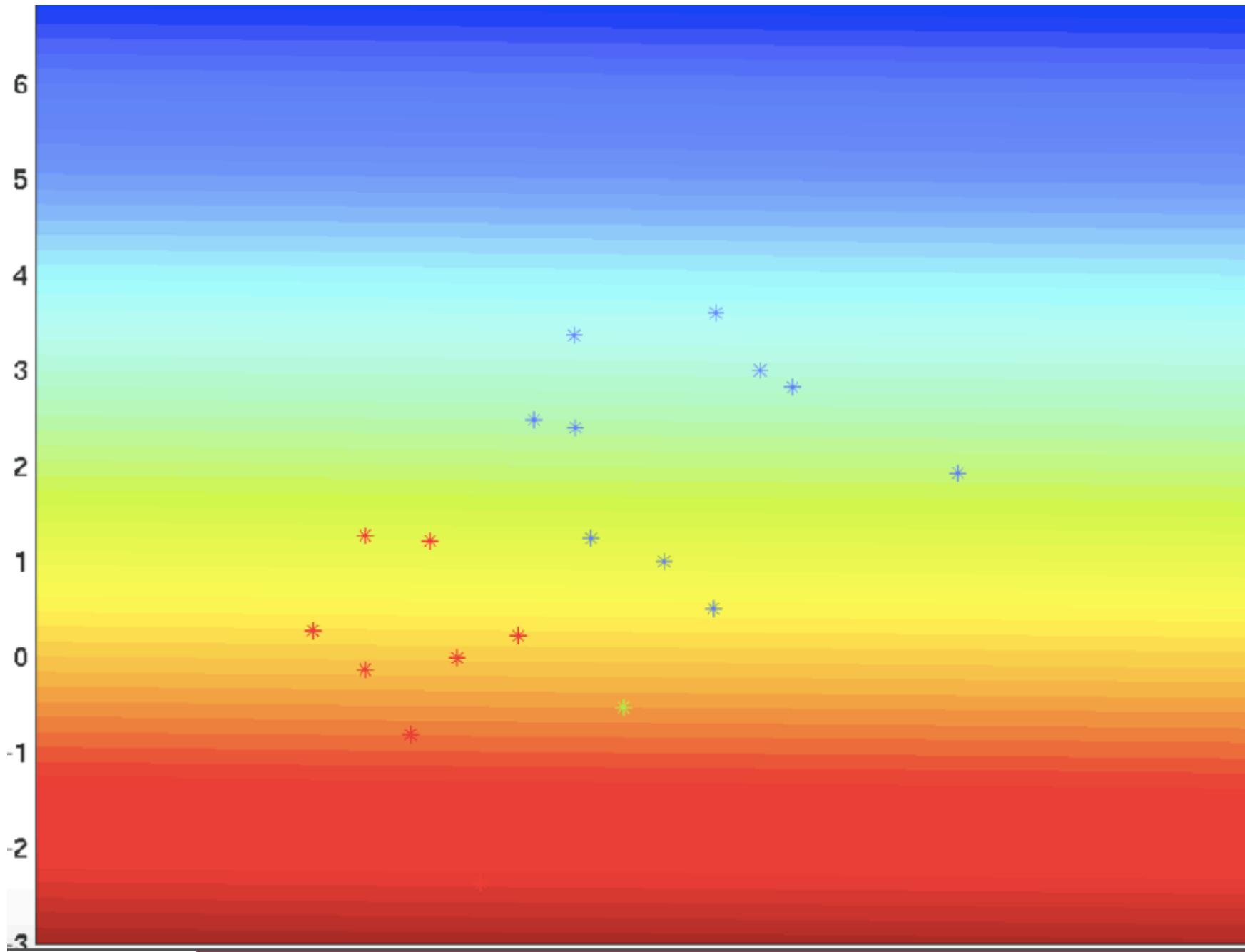
hard

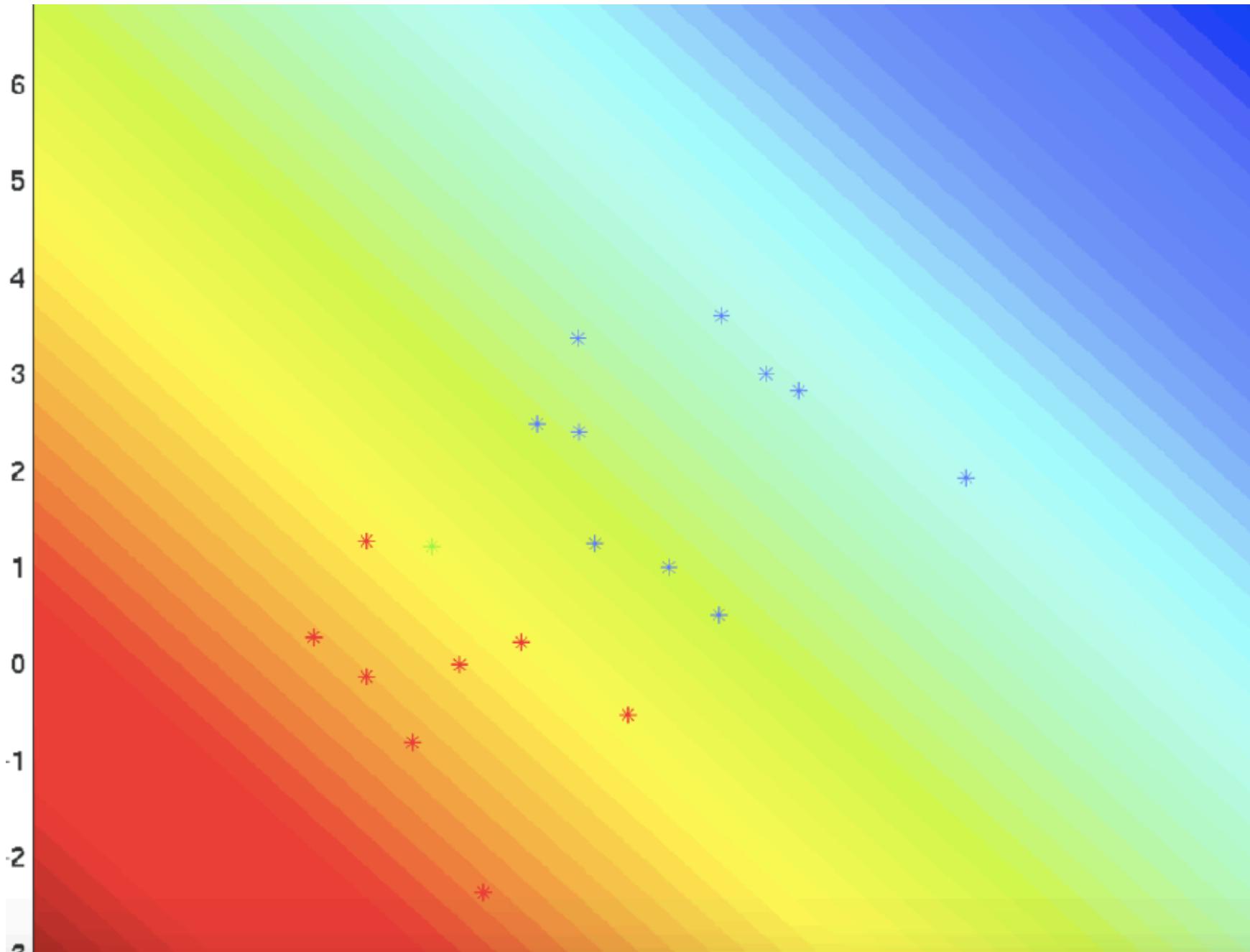


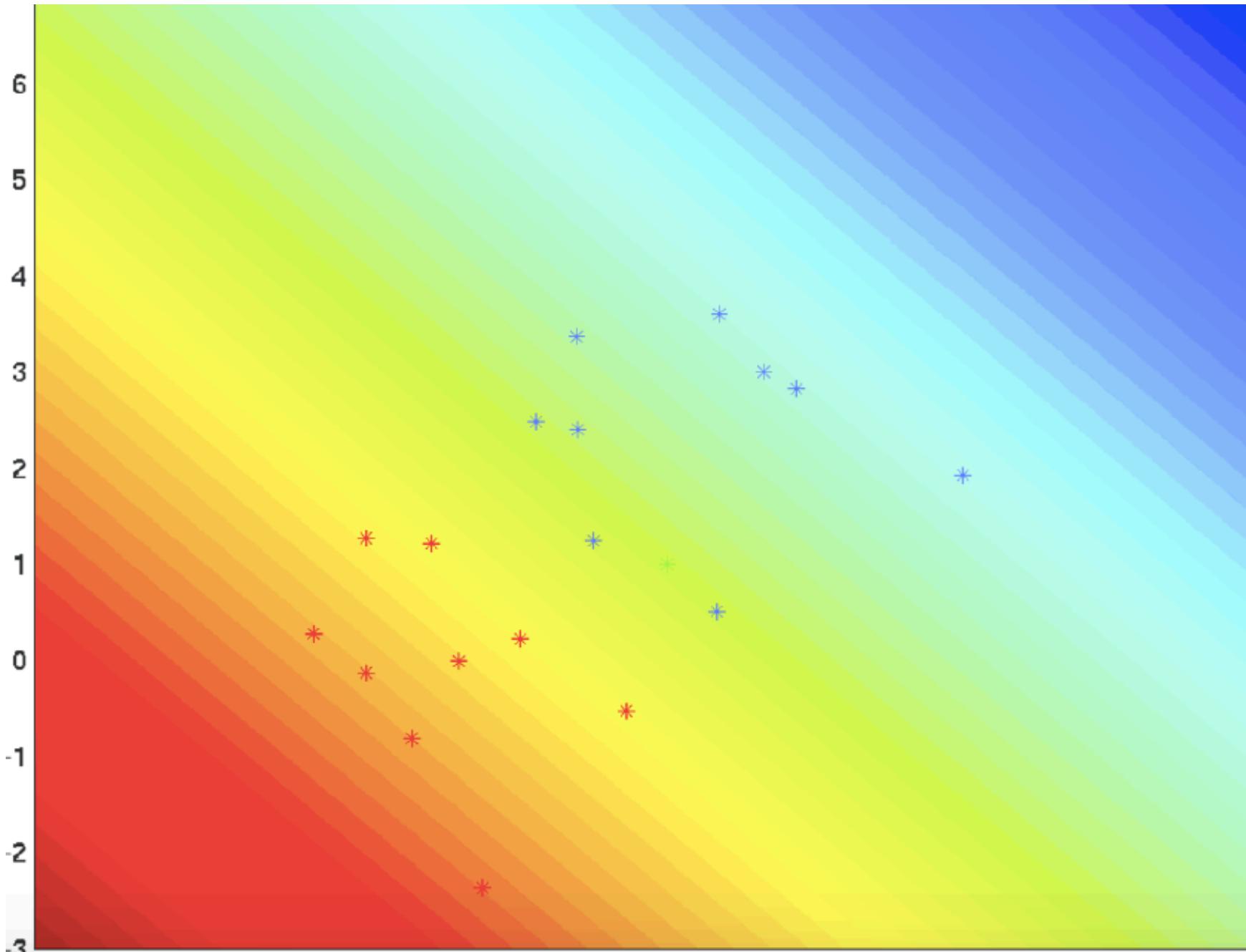
easy











# Concepts & version space

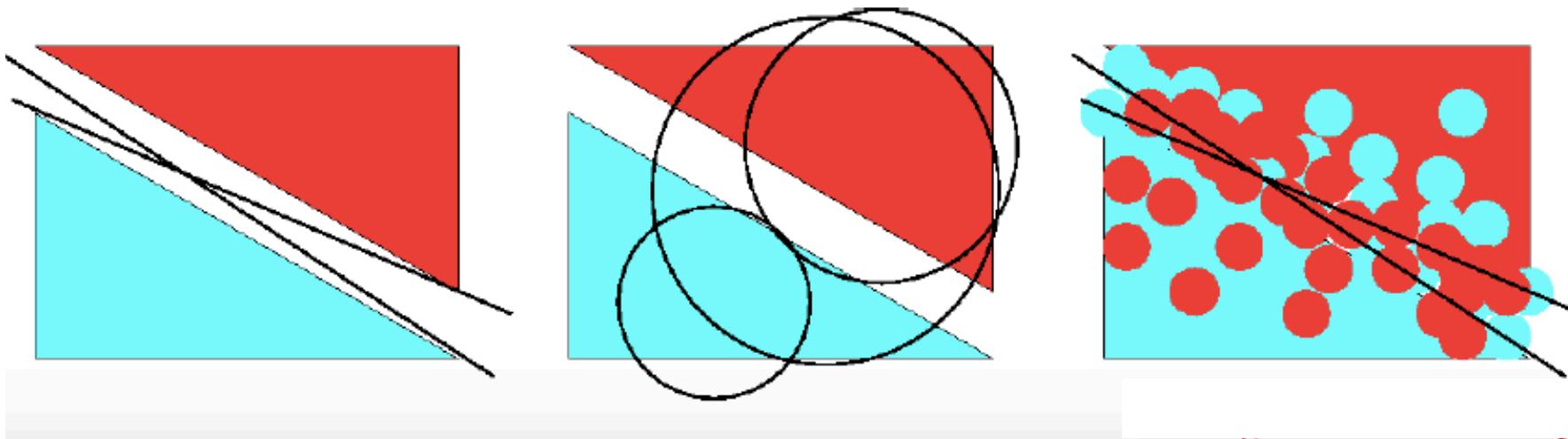
- Realizable concepts
  - Some function exists that can separate data and is included in the concept space
  - For perceptron - data is linearly separable

# Concepts & version space

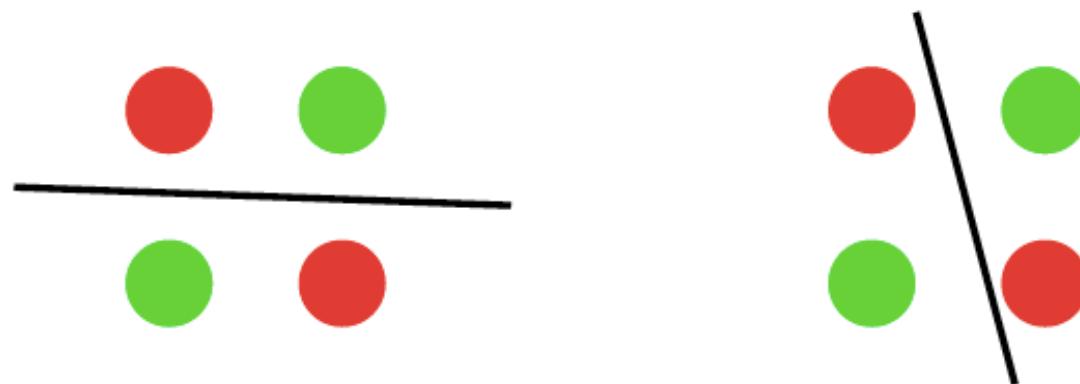
- Realizable concepts
  - Some function exists that can separate data and is included in the concept space
  - For perceptron - data is linearly separable
- Unrealizable concept
  - Data not separable
  - We don't have a suitable function class (often hard to distinguish)

# Concepts & version space

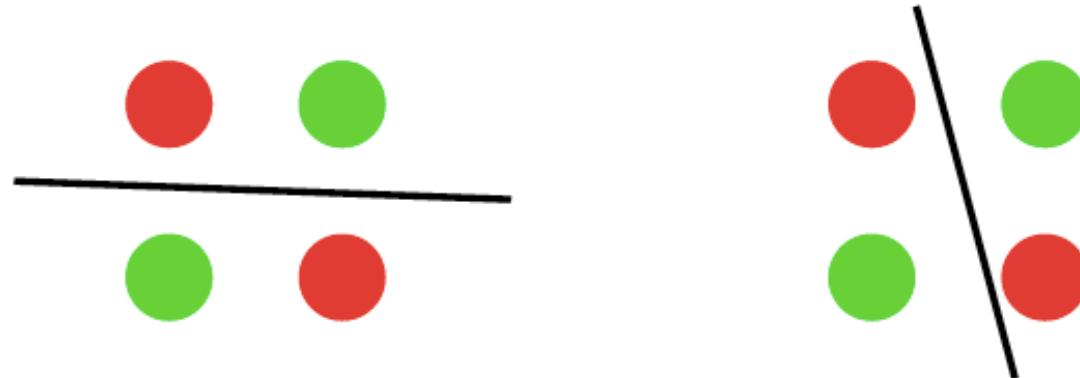
- Realizable concepts
  - Some function exists that can separate data and is included in the concept space
  - For perceptron - data is linearly separable
- Unrealizable concept
  - Data not separable
  - We don't have a suitable function class (often hard to distinguish)



# Minimum error separation

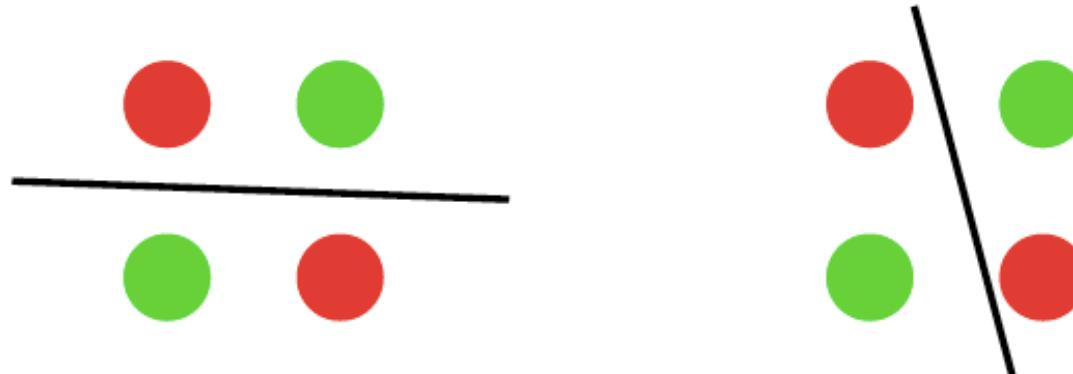


# Minimum error separation



- XOR - not linearly separable
- Nonlinear separation is trivial

# Minimum error separation



- XOR - not linearly separable
- Nonlinear separation is trivial
- Caveat (Minsky & Papert)  
Finding the minimum error linear separator  
is NP hard (this killed Neural Networks in the 70s).

Carnegie Mellon University

**5 minutes break**

# **Non-linearity and preprocessing**

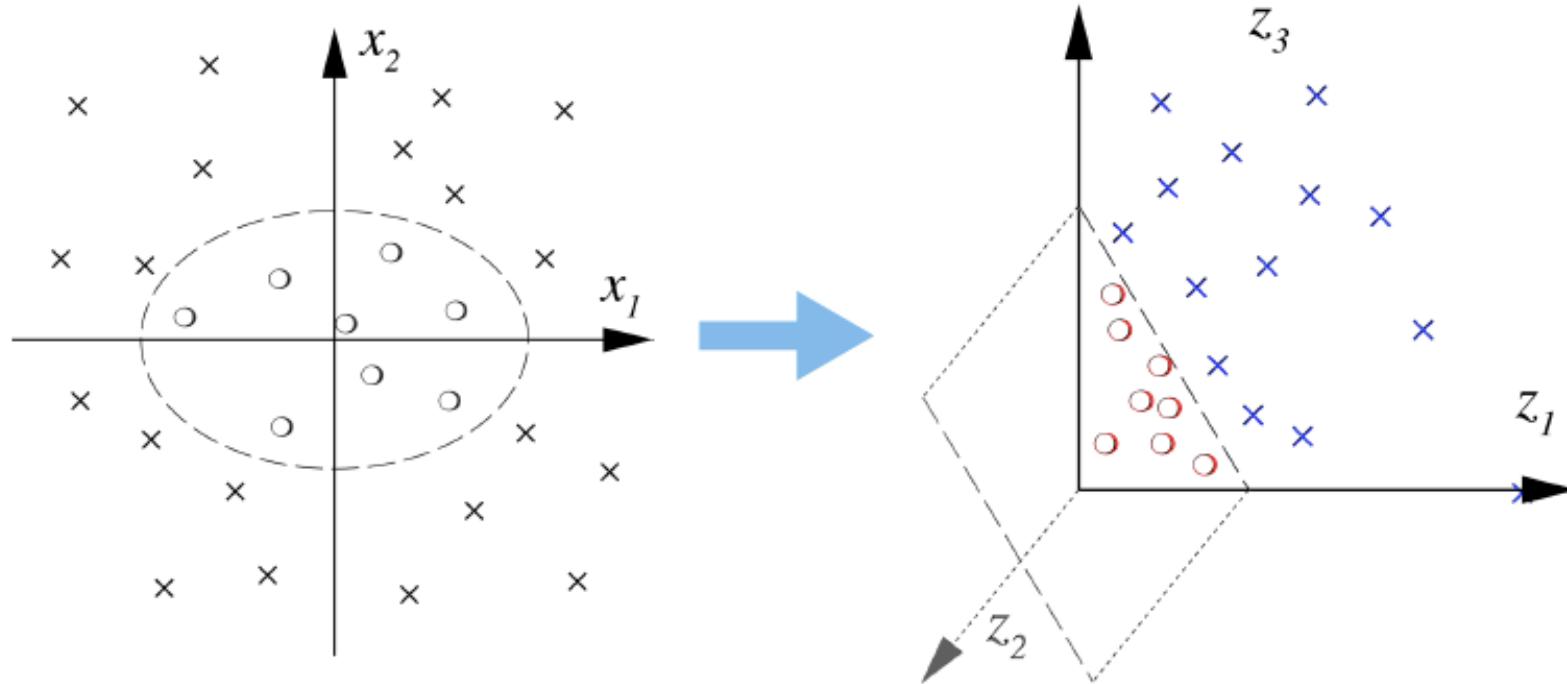
# Nonlinear Features

- Perceptron
  - Map data into feature space  $x \rightarrow \phi(x)$
  - Solve problem in this space
  - Query replace  $\langle x, x' \rangle$  by  $\langle \phi(x), \phi(x') \rangle$  for code

# Nonlinear Features

- Perceptron
  - Map data into feature space  $x \rightarrow \phi(x)$
  - Solve problem in this space
  - Query replace  $\langle x, x' \rangle$  by  $\langle \phi(x), \phi(x') \rangle$  for code
- Feature Perceptron
  - Solution in span of  $\phi(x_i)$

# Quadratic Features



- Separating surfaces are Circles, hyperbolae, parabolae

# Constructing Features (very naive OCR system)

	1	2	3	4	5	6	7	8	9	0
Loops	0	0	0	1	0	1	0	2	1	1
3 Joints	0	0	0	0	0	1	0	0	1	0
4 Joints	0	0	0	1	0	0	0	1	0	0
Angles	0	1	1	1	1	0	1	0	0	0
Ink	1	2	2	2	2	2	1	3	2	2

Delivered-To: [alex.smola@gmail.com](mailto:alex.smola@gmail.com)  
Received: by 10.216.47.73 with SMTP id s5ics361171web;  
Tue, 3 Jan 2012 14:17:53 -0800 (PST)  
Received: by 10.213.17.145 with SMTP id s17mr2519891eba.147.1325629071725;  
Tue, 03 Jan 2012 14:17:51 -0800 (PST)  
Return-Path: <[alex+caf\\_alex.smola@gmail.com@smola.org](mailto:alex+caf_alex.smola@gmail.com@smola.org)>  
Received: from mail-ey0-f175.google.com (mail-ey0-f175.google.com [209.85.215.175])  
by mx.google.com with ESMTPS id n4si29264232ee5.57.2012.01.03.14.17.51  
(version=TLSv1/SSLv3 cipher=OTHER);  
Tue, 03 Jan 2012 14:17:51 -0800 (PST)  
Received-SPF: neutral (google.com: 209.85.215.175 is neither permitted nor denied by best  
guess record for domain of [alex+caf\\_alex.smola@gmail.com@smola.org](mailto:alex+caf_alex.smola@gmail.com@smola.org)) client-  
ip=209.85.215.175;  
Authentication-Results: mx.google.com; spf=neutral (google.com: 209.85.215.175 is neither  
permitted nor denied by best guess record for domain of alex  
[+caf\\_alex.smola@gmail.com@smola.org](mailto:+caf_alex.smola@gmail.com@smola.org)) smtp.mail=[alex+caf\\_alex.smola@gmail.com@smola.org](mailto:alex+caf_alex.smola@gmail.com@smola.org);dkim-pass (test mode) header.i=[@googlemail.com  
Received: by eamll with SMTP id l1so15092746ead.6  
for <\[alex.smola@gmail.com\]\(mailto:alex.smola@gmail.com\)>; Tue, 03 Jan 2012 14:17:51 -0800 \(PST\)  
Received: by 10.205.135.18 with SMTP id ie18mr5325064bkc.72.1325629071362;  
Tue, 03 Jan 2012 14:17:51 -0800 \(PST\)  
X-Forwarded-To: \[alex.smola@gmail.com\]\(mailto:alex.smola@gmail.com\)  
X-Forwarded-For: \[alex@smola.org\]\(mailto:alex@smola.org\) \[alex.smola@gmail.com\]\(mailto:alex.smola@gmail.com\)  
Delivered-To: \[alex@smola.org\]\(mailto:alex@smola.org\)  
Received: by 10.204.65.198 with SMTP id k6cs206093bki;  
Tue, 3 Jan 2012 14:17:50 -0800 \(PST\)  
Received: by 10.52.88.179 with SMTP id bh19mr10729402vdb.38.1325629068795;  
Tue, 03 Jan 2012 14:17:48 -0800 \(PST\)  
Return-Path: <\[alhoff.tim@googlemail.com\]\(mailto:alhoff.tim@googlemail.com\)>  
Received: from mail-vx0-f179.google.com \(mail-vx0-f179.google.com \[209.85.220.179\]\)  
by mx.google.com with ESMTPS id dt4si11767074vdb.93.2012.01.03.14.17.48  
\(version=TLSv1/SSLv3 cipher=OTHER\);  
Tue, 03 Jan 2012 14:17:48 -0800 \(PST\)  
Received-SPF: pass \(google.com: domain of \[alhoff.tim@googlemail.com\]\(mailto:alhoff.tim@googlemail.com\) designates  
209.85.220.179 as permitted sender\) client-ip=209.85.220.179;  
Received: by vcf13 with SMTP id f13so11295098vcb.10  
for <\[alex@smola.org\]\(mailto:alex@smola.org\)>; Tue, 03 Jan 2012 14:17:48 -0800 \(PST\)  
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;  
d=googlemail.com; s=gamma;  
h=mime-version:sender:date:x-google-sender-auth:message-id:subject  
:from:to:content-type;  
bh=WCbZ5sXac25dpH02xCrYD0dts993hKwsAVXpGrFh0w=;  
b=WK2B2+ExWnf/gvTkW6uJvKuP4XeoKn1Jq3USYTm0RARK8dSFjy0QsIHeAP9Yssxp60  
7ngGoIzYqd+ZsyFv0cLAMp1PCJhG8AMcnqNkx0NNMeoFvIp2HQooZwx50Cx5ZRgY+7qX  
uIbbdh41UDXj6UFe16SpLDCptd80Z3gr7+o=  
MIME-Version: 1.0  
Received: by 10.220.108.81 with SMTP id e17mr24104004vcp.67.1325629067787;  
Tue, 03 Jan 2012 14:17:47 -0800 \(PST\)  
Sender: \[alhoff.tim@googlemail.com\]\(mailto:alhoff.tim@googlemail.com\)  
Received: by 10.220.17.129 with HTTP; Tue, 3 Jan 2012 14:17:47 -0800 \(PST\)  
Date: Tue, 3 Jan 2012 14:17:47 -0800  
X-Google-Sender-Auth: 6bw16D17HjZIkx0Eo138NZzyell  
Message-ID: <\[CAF11HDGPBW-SdZg0MWAABIAKydDk9tpem0DijYGjoGO-WC7asg@mail.gmail.com\]\(mailto:CAF11HDGPBW-SdZg0MWAABIAKydDk9tpem0DijYGjoGO-WC7asg@mail.gmail.com\)>  
Subject: CS 281B. Advanced Topics in Learning and Decision Making  
From: Tim Althoff <\[alhoff@eecs.berkeley.edu\]\(mailto:alhoff@eecs.berkeley.edu\)>  
To: \[alex@smola.org\]\(mailto:alex@smola.org\)  
Content-Type: multipart/alternative; boundary=f46d043c7af4b07e8d04b5a7113a  
--f46d043c7af4b07e8d04b5a7113a  
Content-Type: text/plain; charset=ISO-8859-1](mailto:@googlemail.com)

# Feature Engineering for Spam Filtering

- bag of words
- pairs of words
- date & time
- recipient path
- IP number
- sender
- encoding
- links
- ... secret sauce ...

# More feature engineering

- Two Interlocking Spirals  
Transform the data into a radial and angular part
$$(x_1, x_2) = (r \sin \phi, r \cos \phi)$$
- Handwritten Japanese Character Recognition
  - Break down the images into strokes and recognize it
  - Lookup based on stroke order
- Medical Diagnosis
  - Physician's comments
  - Blood status / ECG / height / weight / temperature ...
  - Medical knowledge

# The Perceptron on features

initialize  $w, b = 0$

repeat

    Pick  $(x_i, y_i)$  from data

    if  $y_i(w \cdot \Phi(x_i) + b) \leq 0$  then

$w' = w + y_i\Phi(x_i)$

$b' = b + y_i$

until  $y_i(w \cdot \Phi(x_i) + b) > 0$  for all  $i$



# The Perceptron on features

initialize  $w, b = 0$

repeat

    Pick  $(x_i, y_i)$  from data

    if  $y_i(w \cdot \Phi(x_i) + b) \leq 0$  then

$w' = w + y_i\Phi(x_i)$

$b' = b + y_i$

until  $y_i(w \cdot \Phi(x_i) + b) > 0$  for all  $i$

- Nothing happens if classified correctly
- Weight vector is linear combination  $w = \sum y_i\phi(x_i)$

# The Perceptron on features

initialize  $w, b = 0$

repeat

    Pick  $(x_i, y_i)$  from data

    if  $y_i(w \cdot \Phi(x_i) + b) \leq 0$  then

$w' = w + y_i\Phi(x_i)$

$b' = b + y_i$

until  $y_i(w \cdot \Phi(x_i) + b) > 0$  for all  $i$

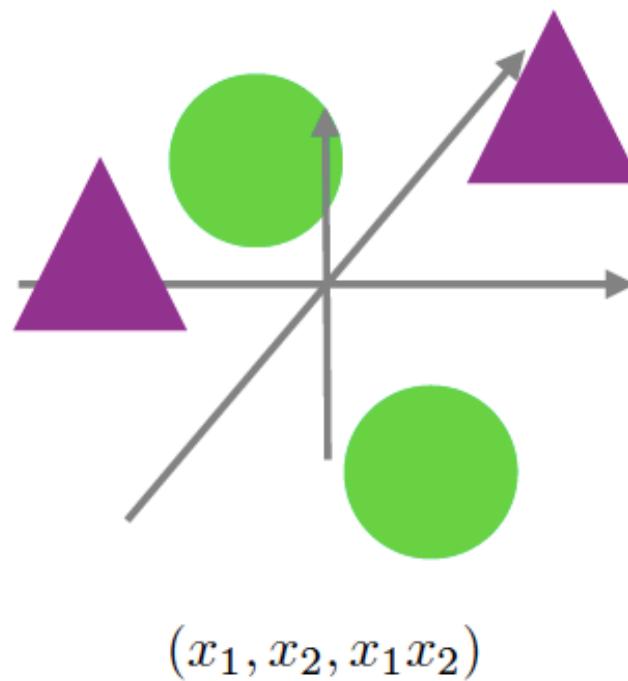
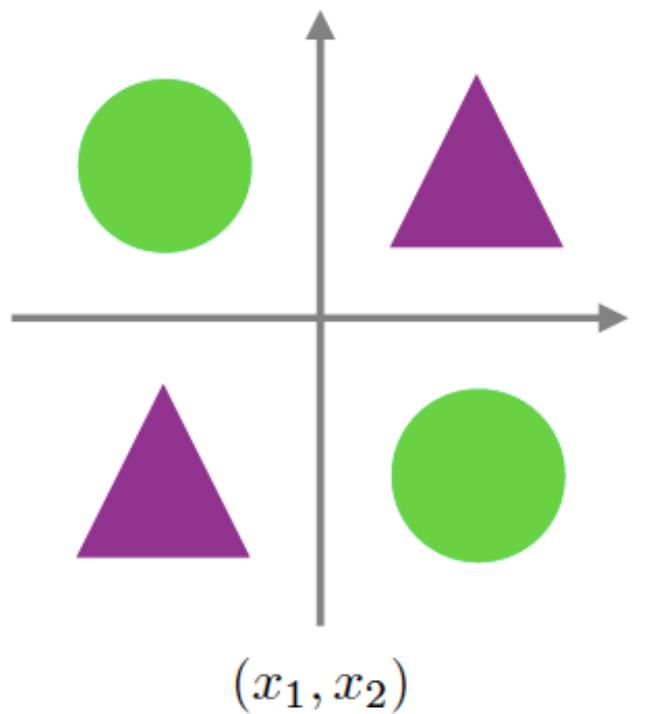
- Nothing happens if classified correctly
- Weight vector is linear combination  $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products  $f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b$

# Problems

- Problems
  - Need domain expert (e.g. Chinese OCR)
  - Often expensive to compute
  - Difficult to transfer engineering knowledge
- Shotgun Solution
  - Compute many features
  - Hope that this contains good ones
  - Do this efficiently

# Kernels

# Solving XOR



- XOR not linearly separable
- Mapping into 3 dimensions makes it easily solvable

# Quadratic Features

**Quadratic Features in  $\mathbb{R}^2$**

$$\Phi(x) := \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right)$$

# Quadratic Features

## Quadratic Features in $\mathbb{R}^2$

$$\Phi(x) := \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right)$$

### Dot Product

$$\begin{aligned}\langle \Phi(x), \Phi(x') \rangle &= \left\langle \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right), \left( {x'_1}^2, \sqrt{2}x'_1x'_2, {x'_2}^2 \right) \right\rangle \\ &= \langle x, x' \rangle^2.\end{aligned}$$

# Quadratic Features

## Quadratic Features in $\mathbb{R}^2$

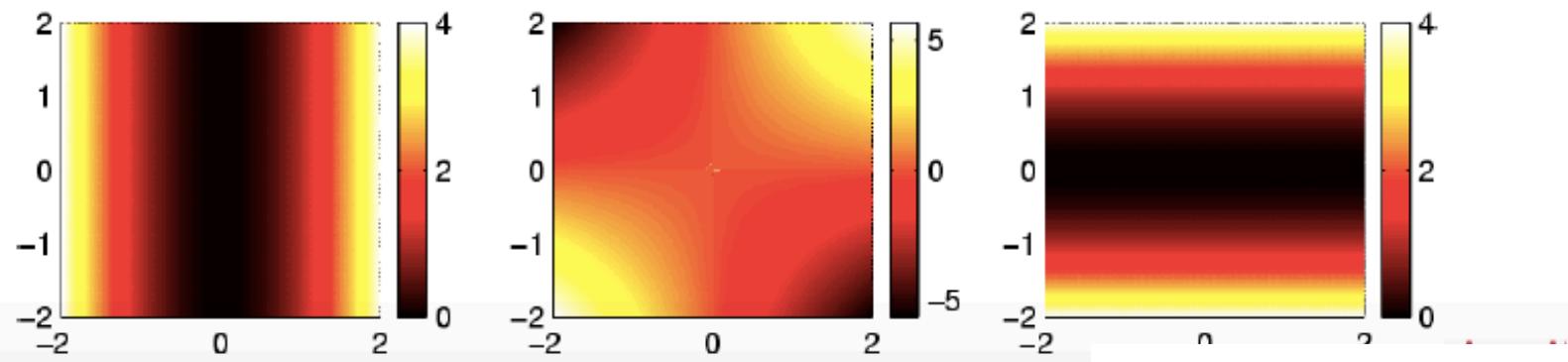
$$\Phi(x) := \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right)$$

### Dot Product

$$\begin{aligned}\langle \Phi(x), \Phi(x') \rangle &= \left\langle \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right), \left( x_1'^2, \sqrt{2}x_1'x_2', x_2'^2 \right) \right\rangle \\ &= \langle x, x' \rangle^2.\end{aligned}$$

### Insight

Trick works for any polynomials of order  $d$  via  $\langle x, x' \rangle^d$ .



# Computational Efficiency

## Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to  $5 \cdot 10^5$  numbers. For higher order polynomial features much worse.

# Computational Efficiency

## Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions.  
This leads to  $5 \cdot 10^5$  numbers. For higher order polynomial features much worse.

## Solution

Don't compute the features, try to compute dot products implicitly. For some features this works . . .

# Computational Efficiency

## Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to  $5 \cdot 10^5$  numbers. For higher order polynomial features much worse.

## Solution

Don't compute the features, try to compute dot products implicitly. For some features this works ...

## Definition

A kernel function  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a symmetric function in its arguments for which the following property holds

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \text{ for some feature map } \Phi.$$

If  $k(x, x')$  is much cheaper to compute than  $\Phi(x)$  ...

# The Kernel Perceptron

initialize  $f = 0$

repeat

    Pick  $(x_i, y_i)$  from data

    if  $y_i f(x_i) \leq 0$  then

$f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i$

until  $y_i f(x_i) > 0$  for all  $i$



# The Kernel Perceptron

initialize  $f = 0$

repeat

Pick  $(x_i, y_i)$  from data

**if**  $y_i f(x_i) \leq 0$  **then**

$f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i$

until  $y_i f(x_i) > 0$  for all  $i$

- Nothing happens if classified correctly
- Weight vector is linear combination  $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products

$$f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b = \sum_{i \in I} y_i k(x_i, x) + b$$

# Polynomial Kernels

## Idea

- We want to extend  $k(x, x') = \langle x, x' \rangle^2$  to  
$$k(x, x') = (\langle x, x' \rangle + c)^d$$
 where  $c > 0$  and  $d \in \mathbb{N}$ .
- Prove that such a kernel corresponds to a dot product.

# Polynomial Kernels

## Idea

- We want to extend  $k(x, x') = \langle x, x' \rangle^2$  to  
$$k(x, x') = (\langle x, x' \rangle + c)^d$$
 where  $c > 0$  and  $d \in \mathbb{N}$ .
- Prove that such a kernel corresponds to a dot product.

## Proof strategy

Simple and straightforward: compute the explicit sum given by the kernel, i.e.

$$k(x, x') = (\langle x, x' \rangle + c)^d = \sum_{i=0}^m \binom{d}{i} (\langle x, x' \rangle)^i c^{d-i}$$

Individual terms  $(\langle x, x' \rangle)^i$  are dot products for some  $\Phi_i(x)$ .

# Kernel Conditions

## Computability

We have to be able to compute  $k(x, x')$  efficiently (much cheaper than dot products themselves).

# Kernel Conditions

## Computability

We have to be able to compute  $k(x, x')$  efficiently (much cheaper than dot products themselves).

## “Nice and Useful” Functions

The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

# Kernel Conditions

## Computability

We have to be able to compute  $k(x, x')$  efficiently (much cheaper than dot products themselves).

## “Nice and Useful” Functions

The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

## Symmetry

Obviously  $k(x, x') = k(x', x)$  due to the symmetry of the dot product  $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$ .

# Kernel Conditions

## Computability

We have to be able to compute  $k(x, x')$  efficiently (much cheaper than dot products themselves).

## “Nice and Useful” Functions

The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

## Symmetry

Obviously  $k(x, x') = k(x', x)$  due to the symmetry of the dot product  $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$ .

## Dot Product in Feature Space

Is there always a  $\Phi$  such that  $k$  really is a dot product?

# Mercer's Theorem

## The Theorem

For any symmetric function  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  which is square integrable in  $\mathcal{X} \times \mathcal{X}$  and which satisfies

$$\int_{\mathcal{X} \times \mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X})$$

there exist  $\phi_i : \mathcal{X} \rightarrow \mathbb{R}$  and numbers  $\lambda_i \geq 0$  where

$$k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x') \text{ for all } x, x' \in \mathcal{X}.$$

# Mercer's Theorem

## The Theorem

For any symmetric function  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  which is square integrable in  $\mathcal{X} \times \mathcal{X}$  and which satisfies

$$\int_{\mathcal{X} \times \mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X})$$

there exist  $\phi_i : \mathcal{X} \rightarrow \mathbb{R}$  and numbers  $\lambda_i \geq 0$  where

$$k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x') \text{ for all } x, x' \in \mathcal{X}.$$

## Interpretation

Double integral is the continuous version of a vector-matrix-vector multiplication. For positive semidefinite matrices we have

$$\sum \sum k(x_i, x_j) \alpha_i \alpha_j \geq 0$$

# Properties

## Distance in Feature Space

Distance between points in feature space via

$$\begin{aligned}d(x, x')^2 &:= \|\Phi(x) - \Phi(x')\|^2 \\&= \langle \Phi(x), \Phi(x) \rangle - 2\langle \Phi(x), \Phi(x') \rangle + \langle \Phi(x'), \Phi(x') \rangle \\&= k(x, x) + k(x', x') - 2k(x, x')\end{aligned}$$



# Properties

## Distance in Feature Space

Distance between points in feature space via

$$\begin{aligned}d(x, x')^2 &:= \|\Phi(x) - \Phi(x')\|^2 \\&= \langle \Phi(x), \Phi(x) \rangle - 2\langle \Phi(x), \Phi(x') \rangle + \langle \Phi(x'), \Phi(x') \rangle \\&= k(x, x) + k(x', x') - 2k(x, x')\end{aligned}$$

## Kernel Matrix

To compare observations we compute dot products, so we study the matrix  $K$  given by

$$K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j)$$

where  $x_i$  are the training patterns.

# Properties

## Distance in Feature Space

Distance between points in feature space via

$$\begin{aligned}d(x, x')^2 &:= \|\Phi(x) - \Phi(x')\|^2 \\&= \langle \Phi(x), \Phi(x) \rangle - 2\langle \Phi(x), \Phi(x') \rangle + \langle \Phi(x'), \Phi(x') \rangle \\&= k(x, x) + k(x', x') - 2k(x, x')\end{aligned}$$

## Kernel Matrix

To compare observations we compute dot products, so we study the matrix  $K$  given by

$$K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j)$$

where  $x_i$  are the training patterns.

## Similarity Measure

The entries  $K_{ij}$  tell us the overlap between  $\Phi(x_i)$  and  $\Phi(x_j)$ , so  $k(x_i, x_j)$  is a similarity measure.

# Properties

## $K$ is Positive Semidefinite

Claim:  $\alpha^\top K \alpha \geq 0$  for all  $\alpha \in \mathbb{R}^m$  and all kernel matrices  $K \in \mathbb{R}^{m \times m}$ . Proof:

# Properties

## $K$ is Positive Semidefinite

Claim:  $\alpha^\top K \alpha \geq 0$  for all  $\alpha \in \mathbb{R}^m$  and all kernel matrices  $K \in \mathbb{R}^{m \times m}$ . Proof:

$$\begin{aligned}\sum_{i,j}^m \alpha_i \alpha_j K_{ij} &= \sum_{i,j}^m \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle \\ &= \left\langle \sum_i^m \alpha_i \Phi(x_i), \sum_j^m \alpha_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^m \alpha_i \Phi(x_i) \right\|^2\end{aligned}$$

# Properties

## $K$ is Positive Semidefinite

Claim:  $\alpha^\top K \alpha \geq 0$  for all  $\alpha \in \mathbb{R}^m$  and all kernel matrices  $K \in \mathbb{R}^{m \times m}$ . Proof:

$$\begin{aligned}\sum_{i,j}^m \alpha_i \alpha_j K_{ij} &= \sum_{i,j}^m \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle \\ &= \left\langle \sum_i^m \alpha_i \Phi(x_i), \sum_j^m \alpha_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^m \alpha_i \Phi(x_i) \right\|^2\end{aligned}$$

## Kernel Expansion

If  $w$  is given by a linear combination of  $\Phi(x_i)$  we get

$$\langle w, \Phi(x) \rangle = \left\langle \sum_{i=1}^m \alpha_i \Phi(x_i), \Phi(x) \right\rangle = \sum_{i=1}^m \alpha_i k(x_i, x).$$

# A Counterexample

## A Candidate for a Kernel

$$k(x, x') = \begin{cases} 1 & \text{if } \|x - x'\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

This is symmetric and gives us some information about the proximity of points, yet it is not a proper kernel . . .

# A Counterexample

## A Candidate for a Kernel

$$k(x, x') = \begin{cases} 1 & \text{if } \|x - x'\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

This is symmetric and gives us some information about the proximity of points, yet it is not a proper kernel . . .

## Kernel Matrix

We use three points,  $x_1 = 1, x_2 = 2, x_3 = 3$  and compute the resulting “kernelmatrix”  $K$ . This yields

$$K = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \text{ and eigenvalues } (\sqrt{2}-1)^{-1}, 1 \text{ and } (1-\sqrt{2}).$$

as eigensystem. Hence  $k$  is not a kernel.

# Examples

## Examples of kernels $k(x, x')$

Linear

$$\langle x, x' \rangle$$

Laplacian RBF

$$\exp(-\lambda \|x - x'\|)$$

Gaussian RBF

$$\exp(-\lambda \|x - x'\|^2)$$

Polynomial

$$(\langle x, x' \rangle + c)^d, c \geq 0, d \in \mathbb{N}$$

B-Spline

$$B_{2n+1}(x - x')$$

Cond. Expectation

$$\mathbf{E}_c[p(x|c)p(x'|c)]$$

# Examples

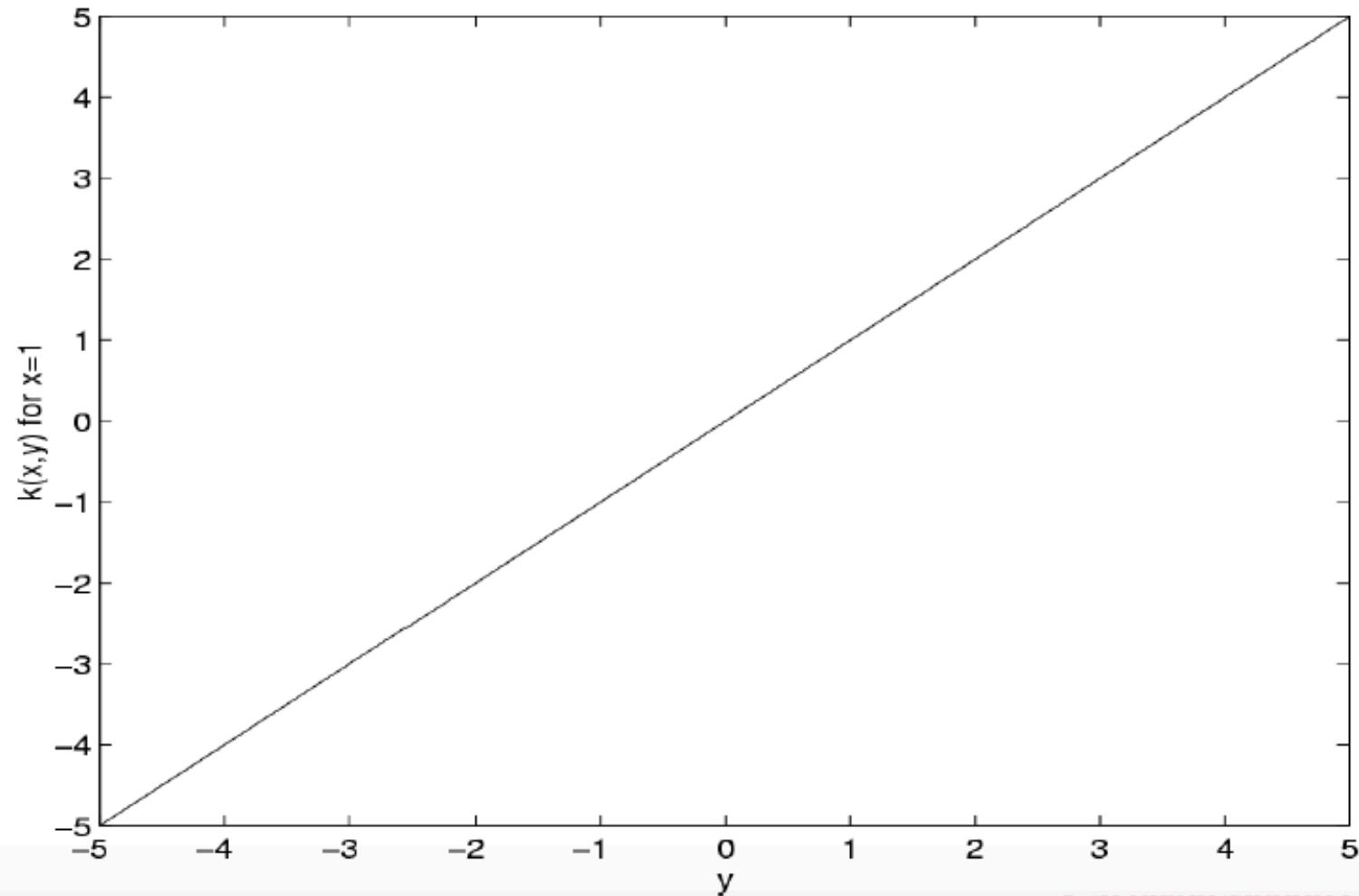
## Examples of kernels $k(x, x')$

Linear	$\langle x, x' \rangle$
Laplacian RBF	$\exp(-\lambda \ x - x'\ )$
Gaussian RBF	$\exp(-\lambda \ x - x'\ ^2)$
Polynomial	$(\langle x, x' \rangle + c)^d, c \geq 0, d \in \mathbb{N}$
B-Spline	$B_{2n+1}(x - x')$
Cond. Expectation	$E_c[p(x c)p(x' c)]$

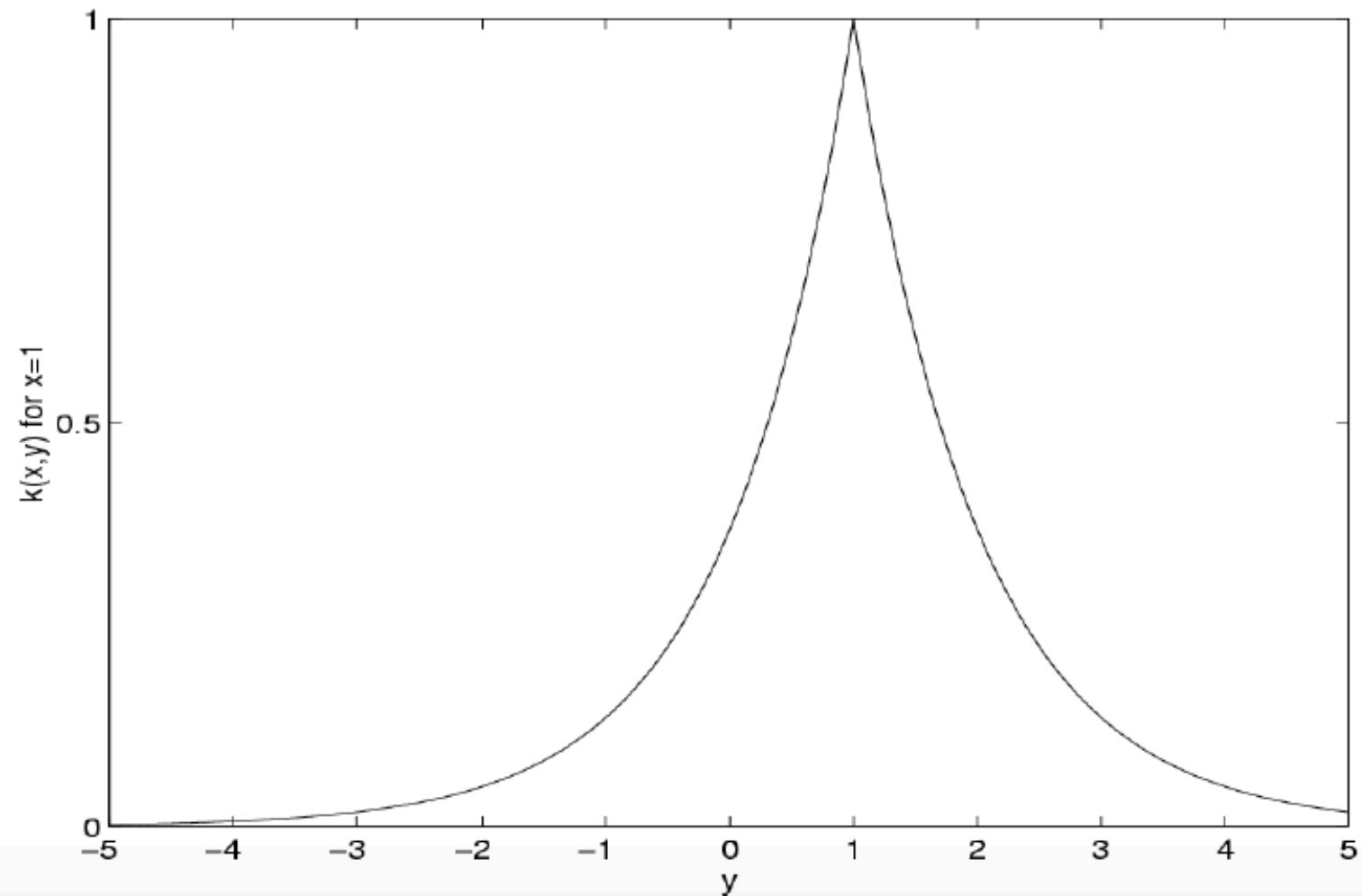
## Simple trick for checking Mercer's condition

Compute the Fourier transform of the kernel and check that it is nonnegative.

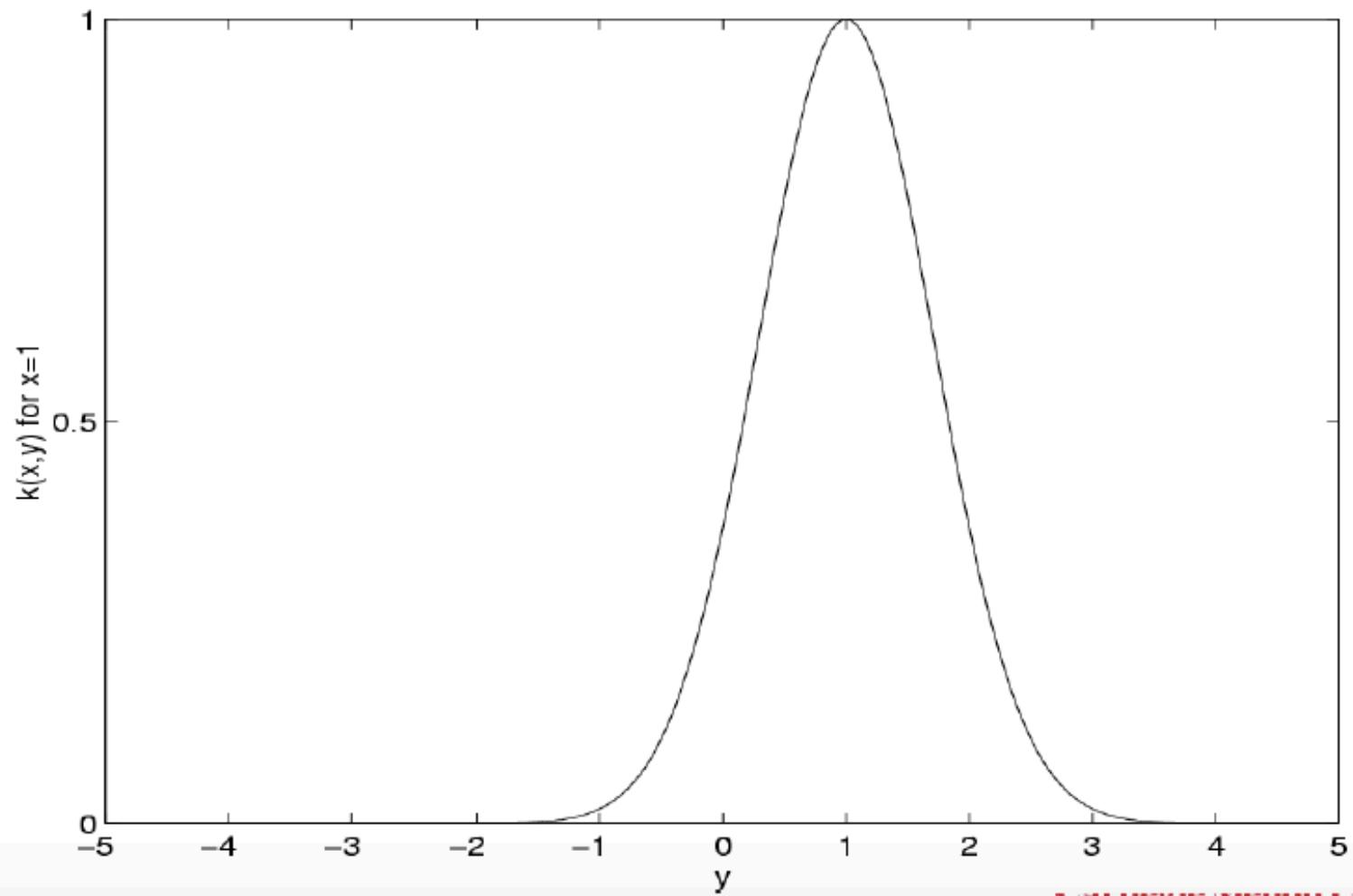
# Linear Kernel



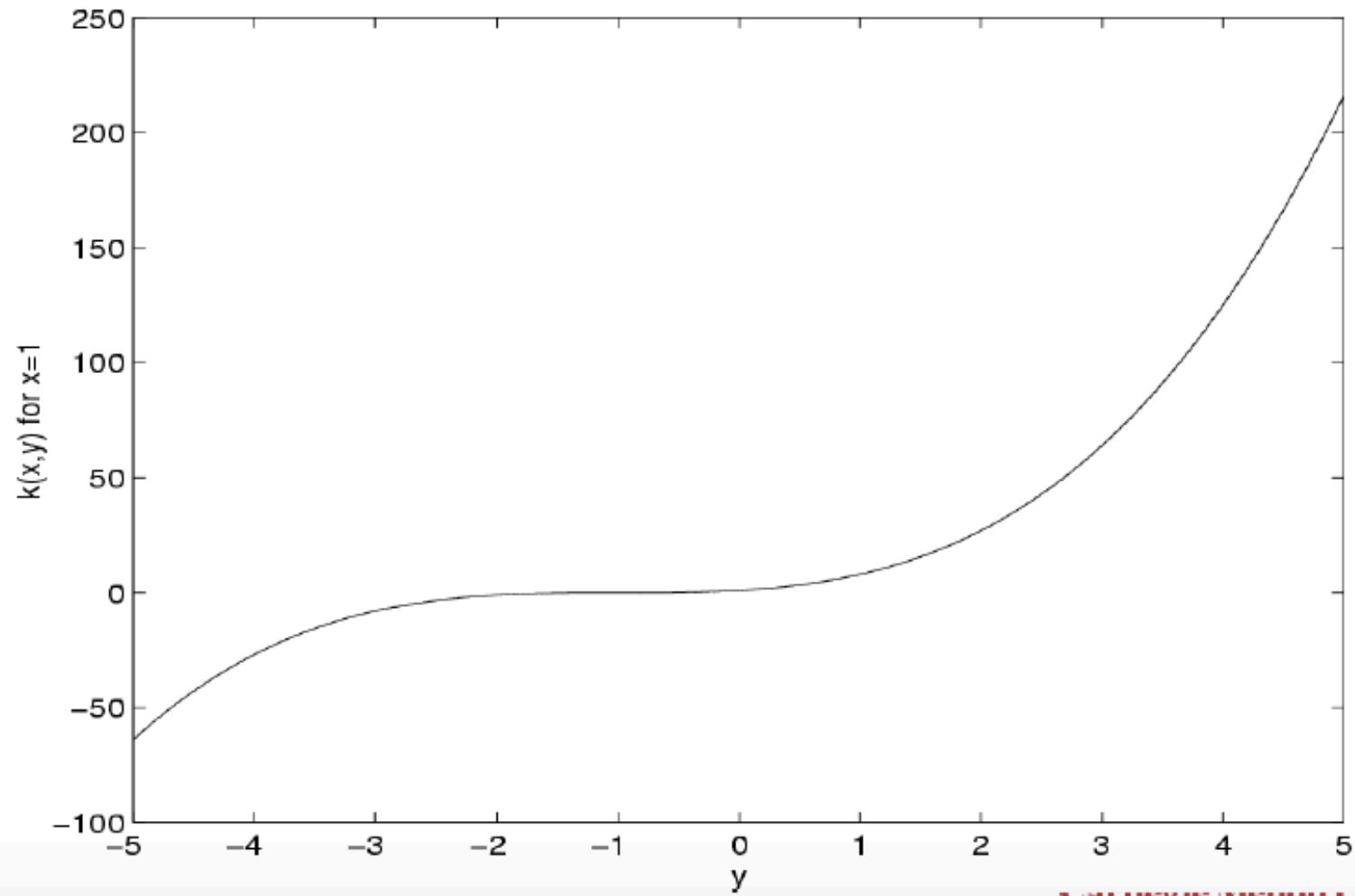
# Laplacian Kernel



# Gaussian Kernel

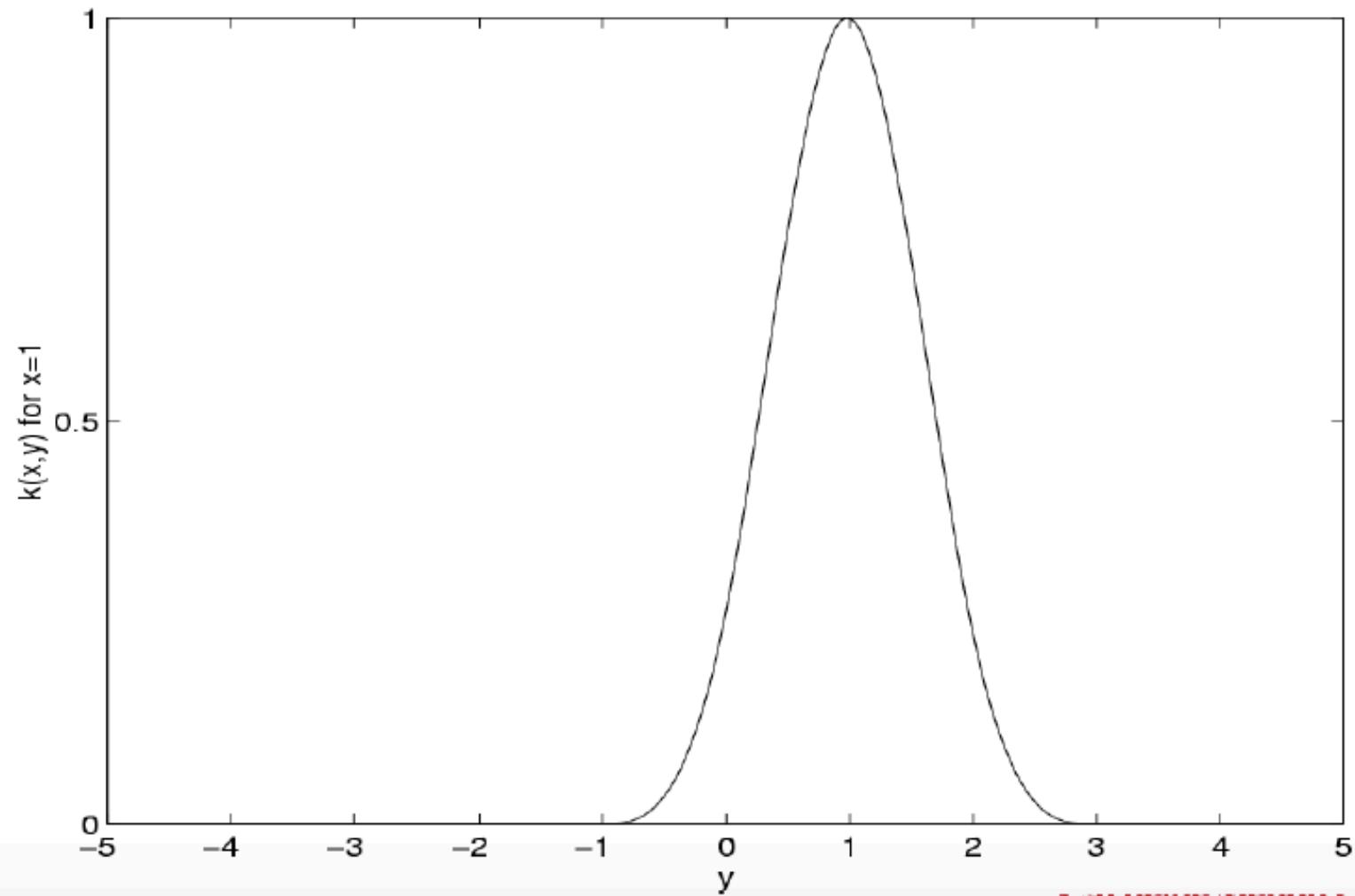


# Polynomial of order 3



Carnegie Mellon University

# B<sub>3</sub> Spline Kernel



**That's all!**