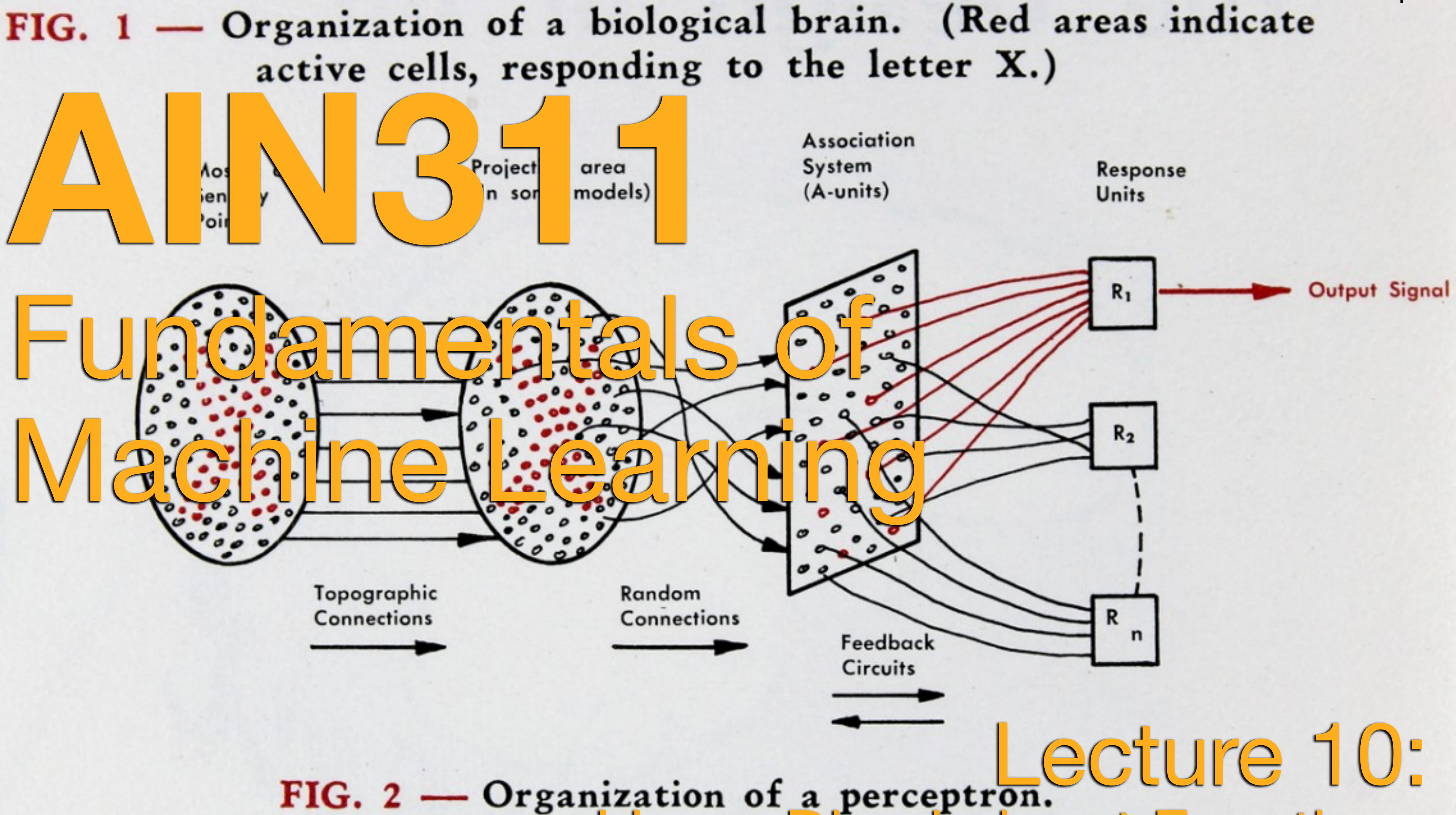
Illustration: Frank Rosenblatt's Perceptron



AIN311

Fundamentals of Machine Learning

Lecture 10: Linear Discriminant Functions Perceptron

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Fall 2024

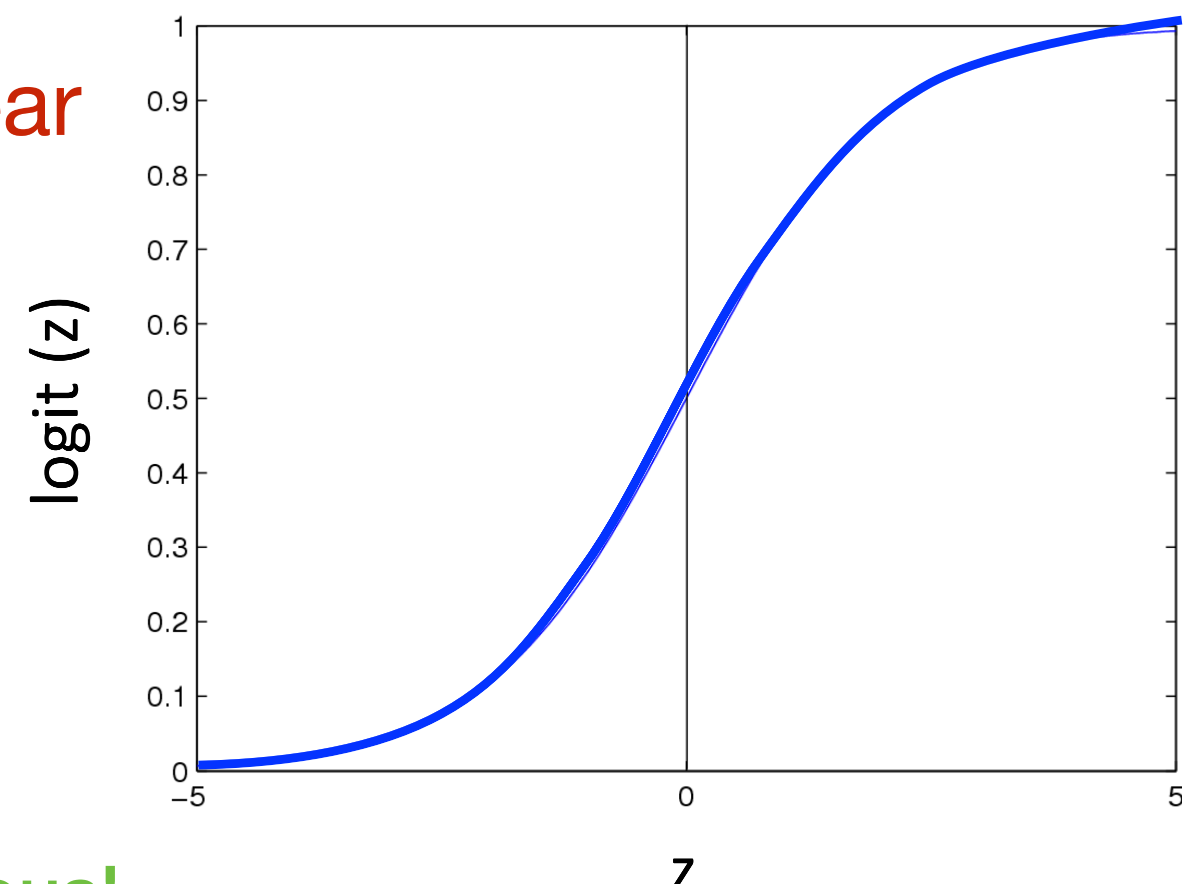
Last time… **Logistic Regression**

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Logistic function applied to linear function of the data

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



Features can be discrete or continuous!

slide by Aarti Singh & Barnabás Póczos

Last time.. **Logistic Regression vs. Gaussian Naïve Bayes**

- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by maximizing conditional likelihood
  - no closed-form solution
  - concave ! global optimum with gradient ascent
- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class! assumption on P(**X**|Y)
  - LR: Functional form of P(Y|**X**), no assumption on P(**X**|Y)
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit

slide by Aarti Singh & Barnabás Póczos

Linear Discriminant Functions

Linear Discriminant Function

- Linear discriminant function for a vector $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

where $\mathbf{w}$ is called weight vector, and w_0 is a bias.

- The classification function is

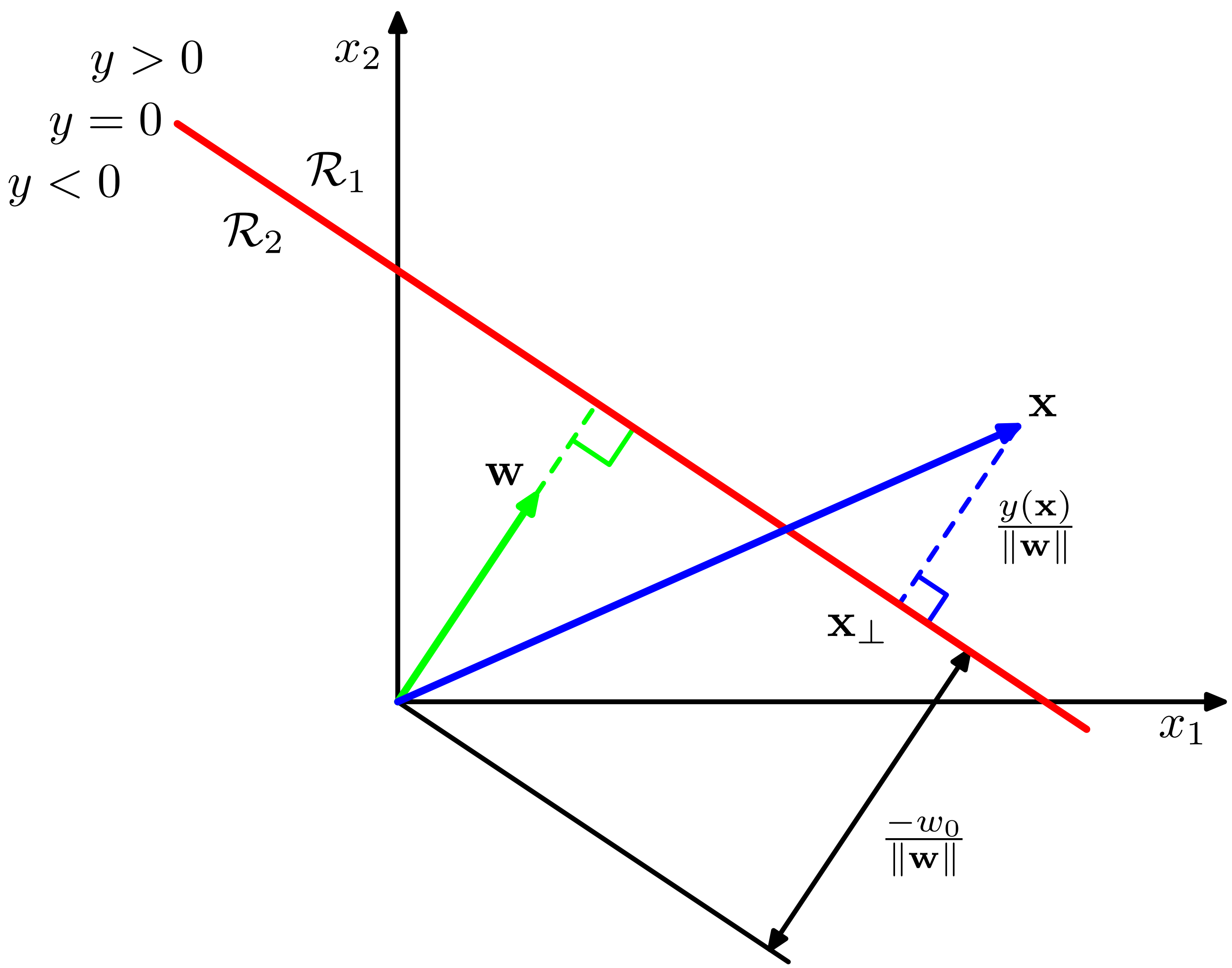
$$C(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T \mathbf{x} + w_0)$$

where step function sign(·) is defined as

$$\mathrm{sign}(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases}$$

slide by Ce Liu

Properties of Linear Discriminant Functions



- The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0.
- The signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$
- $y(\mathbf{x})$ gives a signed measure of the perpendicular distance r of the point $\mathbf{x}$ from the decision surface

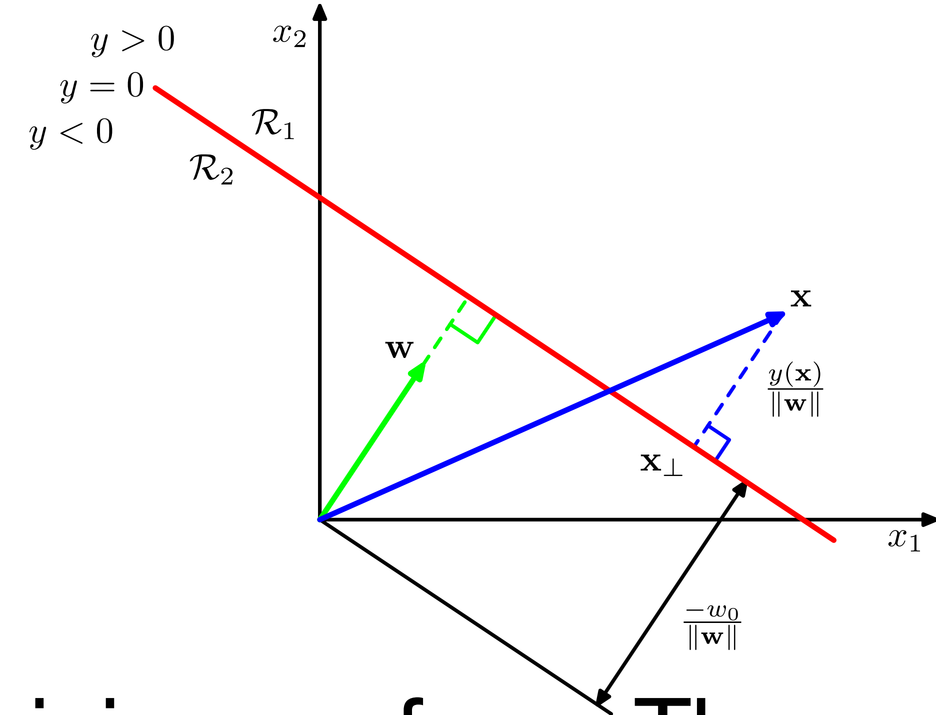
- $y(\mathbf{x}) = 0$ for $\mathbf{x}$ on the decision surface. The normal distance from the origin to the decision surface is

$$\frac{\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

- So w_0 determines the location of the decision surface.

slide by Ce Liu

Properties of Linear Discriminant Functions



- Let

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

where $\mathbf{x}_\perp$ is the projection $\mathbf{x}$ on the decision surface. Then

$$\mathbf{w}^T\mathbf{x} = \mathbf{w}^T\mathbf{x}_\perp + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|}$$

$$\mathbf{w}^T\mathbf{x} + w_0 = \mathbf{w}^T\mathbf{x}_\perp + w_0 + r\|\mathbf{w}\|$$

$$y(\mathbf{x}) = r\|\mathbf{w}\|$$

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

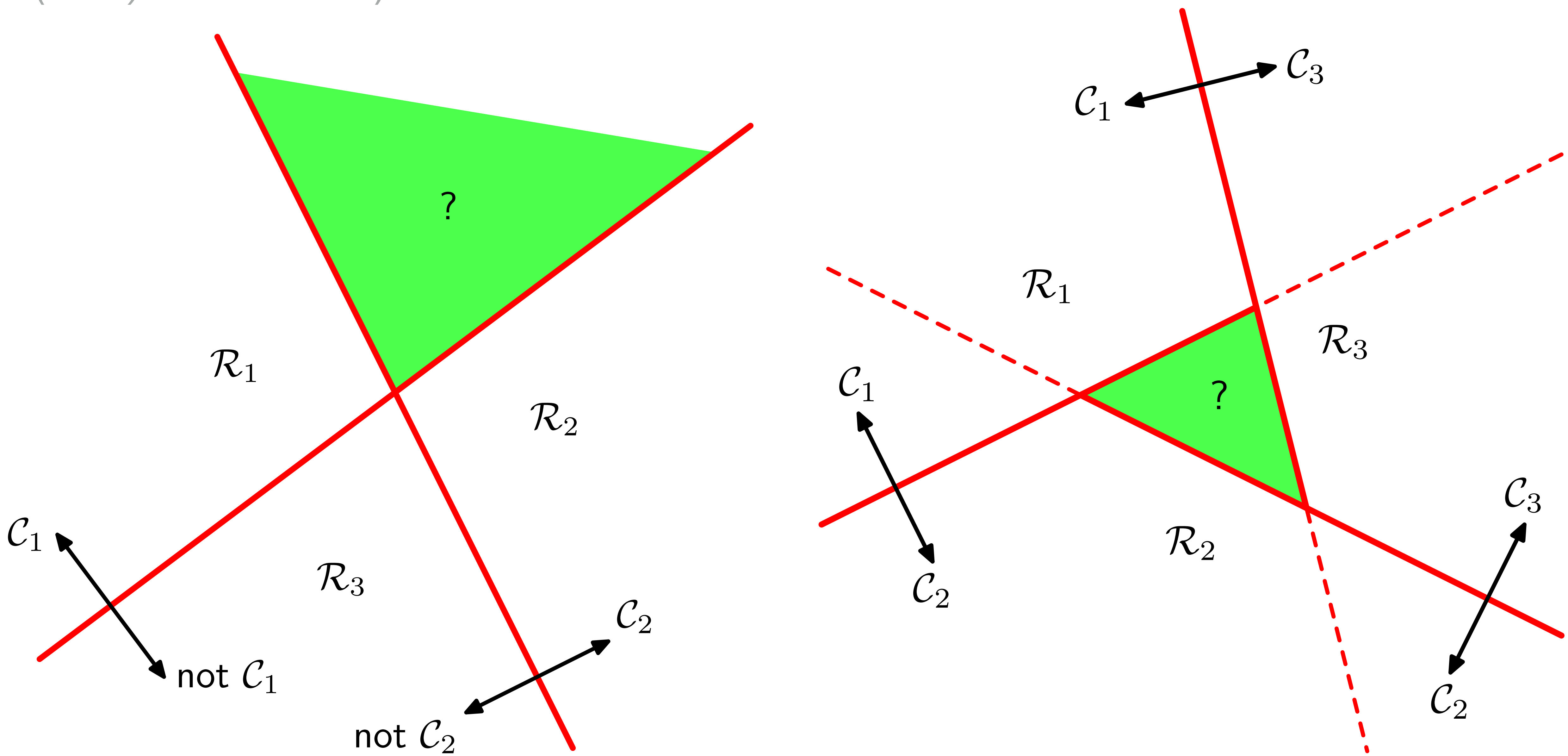
- Simpler notion: define $\widetilde{\mathbf{w}} = (w_0, \mathbf{w})$ and $\widetilde{\mathbf{x}} = (1, \mathbf{x})$ so that

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^T\widetilde{\mathbf{x}}$$

slide by Ce Liu

Multiple Classes: Simple Extension

- **One-versus-the-rest** classifier: classify C_k and samples not in C_k. *(K –1 classifiers)*
- **One-versus-one** classifier: classify every pair of classes. K*(K –1)/2 classifiers*)



slide by Ce Liu

Multiple Classes: K-Class Discriminant

- A single *K*-class discriminant comprising *K* linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- Decision function

$$\mathcal{C}(\mathbf{x}) = k, \text{ if } y_k(\mathbf{x}) > y_j(\mathbf{x}) \; \forall \, j \neq k$$

- The decision boundary between class C_k and C_j is given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

slide by Ce Liu

Property of the Decision Regions

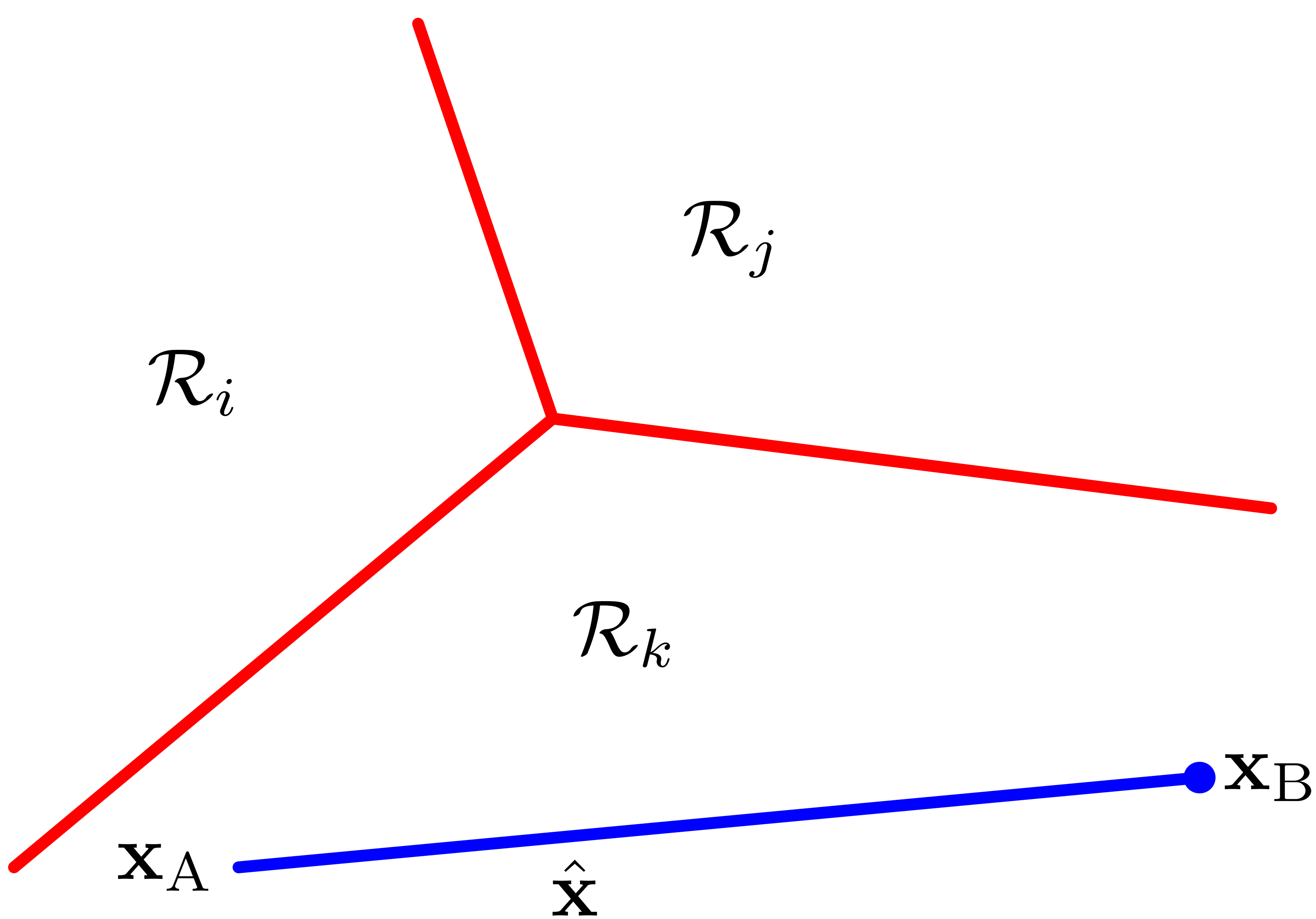
Theorem

The decision regions of the K-class discriminant $y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}$ are singly connected and convex.

Property of the Decision Regions

Theorem

The decision regions of the K-class discriminant $y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}$ are singly connected and convex.



If two points $\mathbf{x}_\mathrm{A}$ and $\mathbf{x}_\mathrm{B}$ both lie inside the same decision region R_k, then any point $\mathbf{x}$ that lies on the line connecting these two points must also lie in R_k, and hence the decision region must be singly connected and convex.

slide by Ce Liu

Fisher's Linear Discriminant

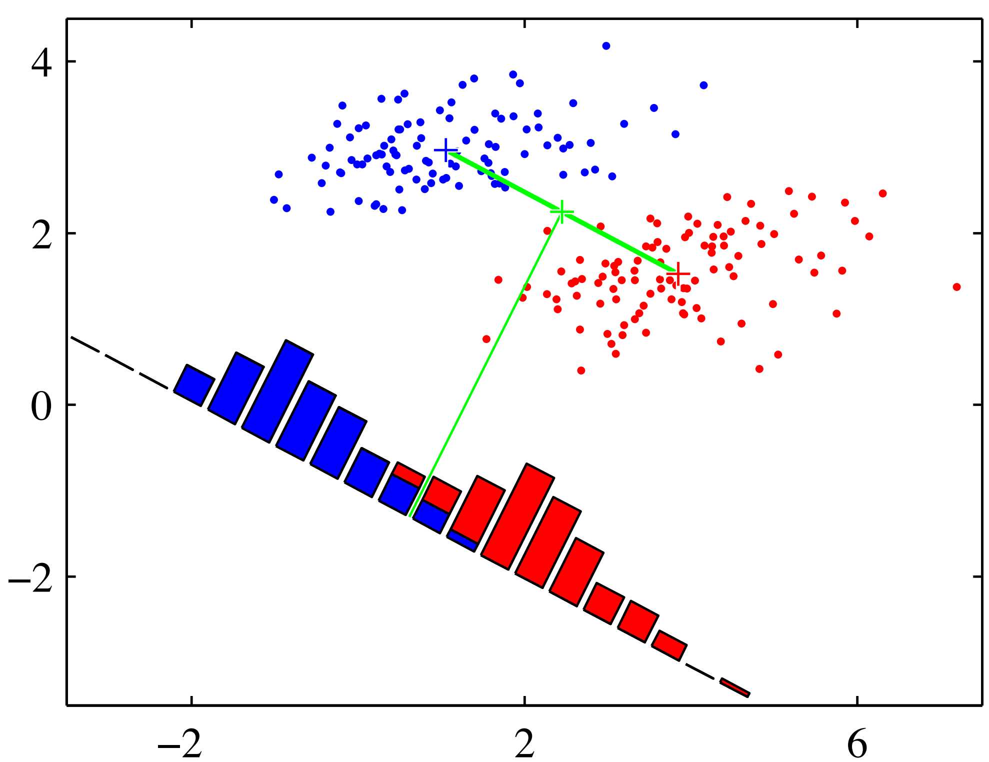
- Pursue the optimal linear projection on which the two classes can be maximally separated

$$y = \mathbf{w}^T \mathbf{x}$$

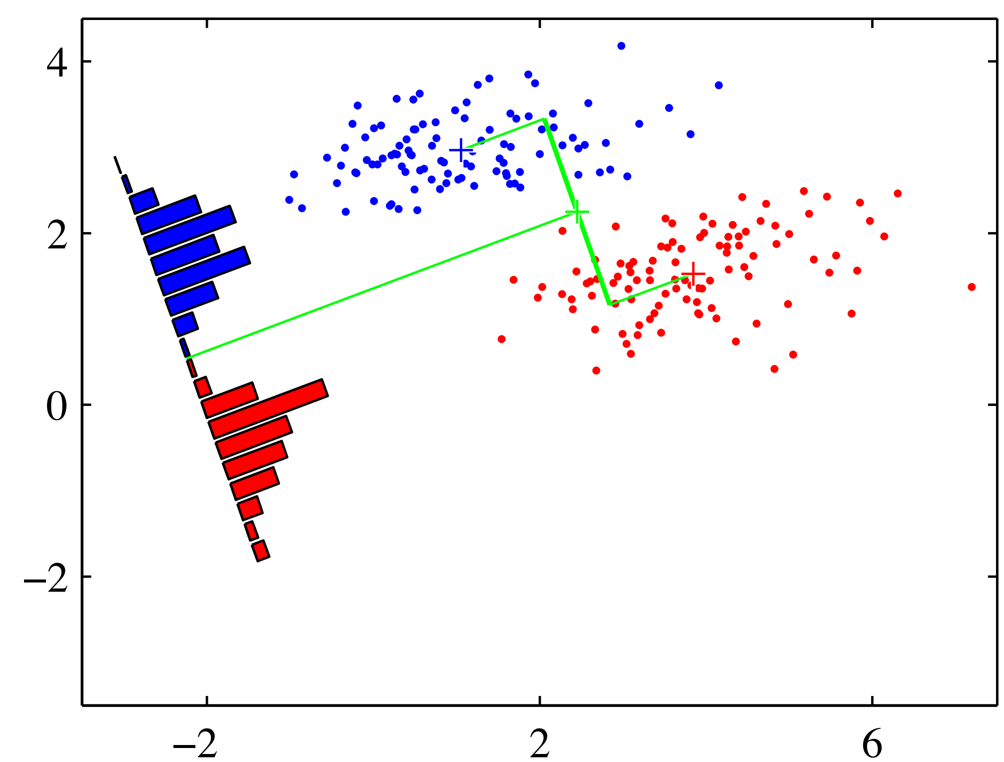
- The mean vectors of the two classes

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

A way to view a linear classification model is in terms of dimensionality reduction.



Difference of means



Fisher's Linear Discriminant

slide by Ce Liu

What's a Good Projection?

- After projection, the two classes are separated as much as possible. Measured by the distance between projected center

$$\left(\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\right)^2 = \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w}$$
$$= \mathbf{w}^T\mathbf{S}_B\mathbf{w}$$

where $\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$ is called **between-class** covariance matrix.

- After projection, the variances of the two classes are as small as possible. Measured by the within-class covariance

$$\mathbf{w}^T\mathbf{S}_W\mathbf{w}$$

where

$$\mathbf{S}_W = \sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

slide by Ce Liu

Fisher's Linear Discriminant

- Fisher criterion: maximize the ratio w.r.t. $\mathbf{w}$

$$J(\mathbf{w}) = \frac{\texttt{Between-class variance}}{\texttt{Within-class variance}} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- Recall the quotient rule: for $f(x) = \frac{g(x)}{h(x)}$

$$f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{h^2(x)}$$

- Setting $\nabla J(\boldsymbol{w}) = 0$, we obtain

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$$

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) (\mathbf{m}_2 - \mathbf{m}_1) \Big((\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w} \Big)$$

- Terms $\mathbf{w}^T\mathbf{S}_B\mathbf{w}$, $\mathbf{w}^T\mathbf{S}_W\mathbf{w}$ and $(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w}$ are scalars, and we only care about directions. So the scalars are dropped. Therefore

$$\mathbf{w} \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

slide by Ce Liu

From Fisher's Linear Discriminant to Classifiers

- Fisher's Linear Discriminant is not a classifier; it only decides on an optimal projection to convert high-dimensional classification problem to 1D.

- A bias (threshold) is needed to form a linear classifier (multiple thresholds lead to nonlinear classifiers). The final classifier has the form

$$y(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + w_0)$$

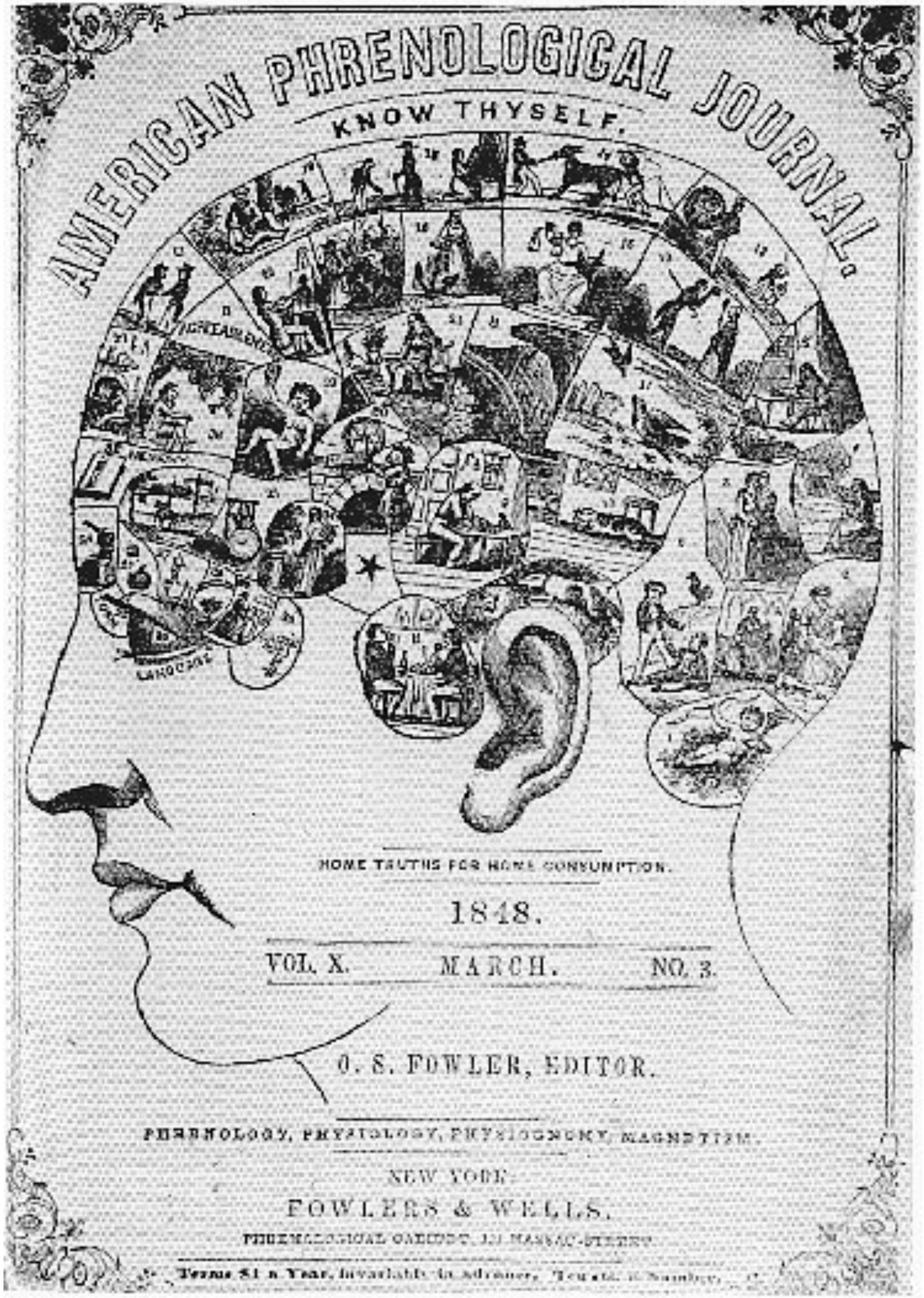
  where the nonlinear activation function sign(·) is a step function

$$\mathrm{sign}(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases}$$

- How to decide the bias w_0?

slide by Ce Liu

Perceptron



early theories
of the brain

slide by Alex Smola

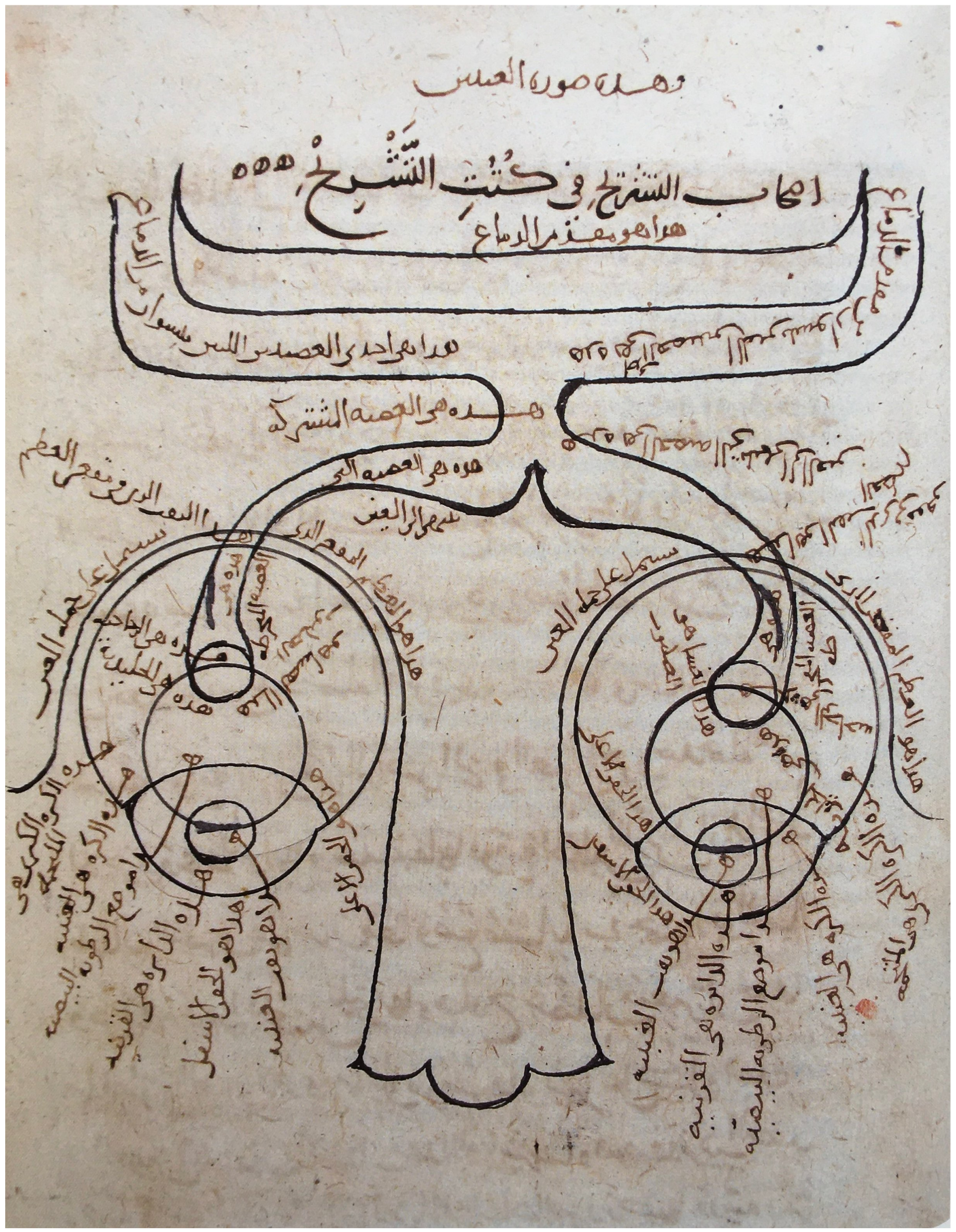
Biology and Learning

- 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.

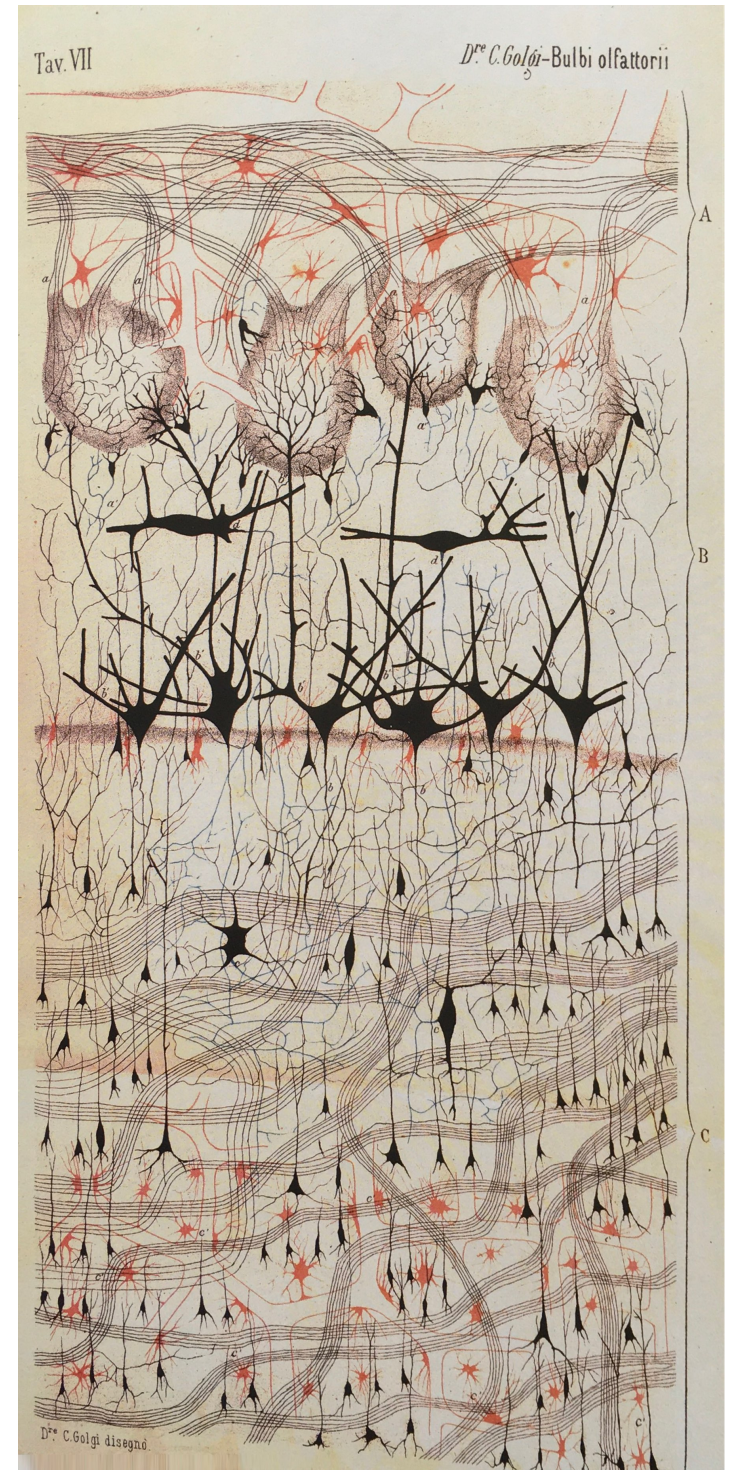
- Training mechanisms
  - Behavioral modification of individuals (learning)
    Successful behavior is rewarded (e.g. food).
  - Hard-coded behavior in the genes (instinct)
    The wrongly coded animal does not reproduce.

slide by Alex Smola

Nervous System



The oldest known drawing of the nervous system by Ibn al-Haytham (published in 1083)

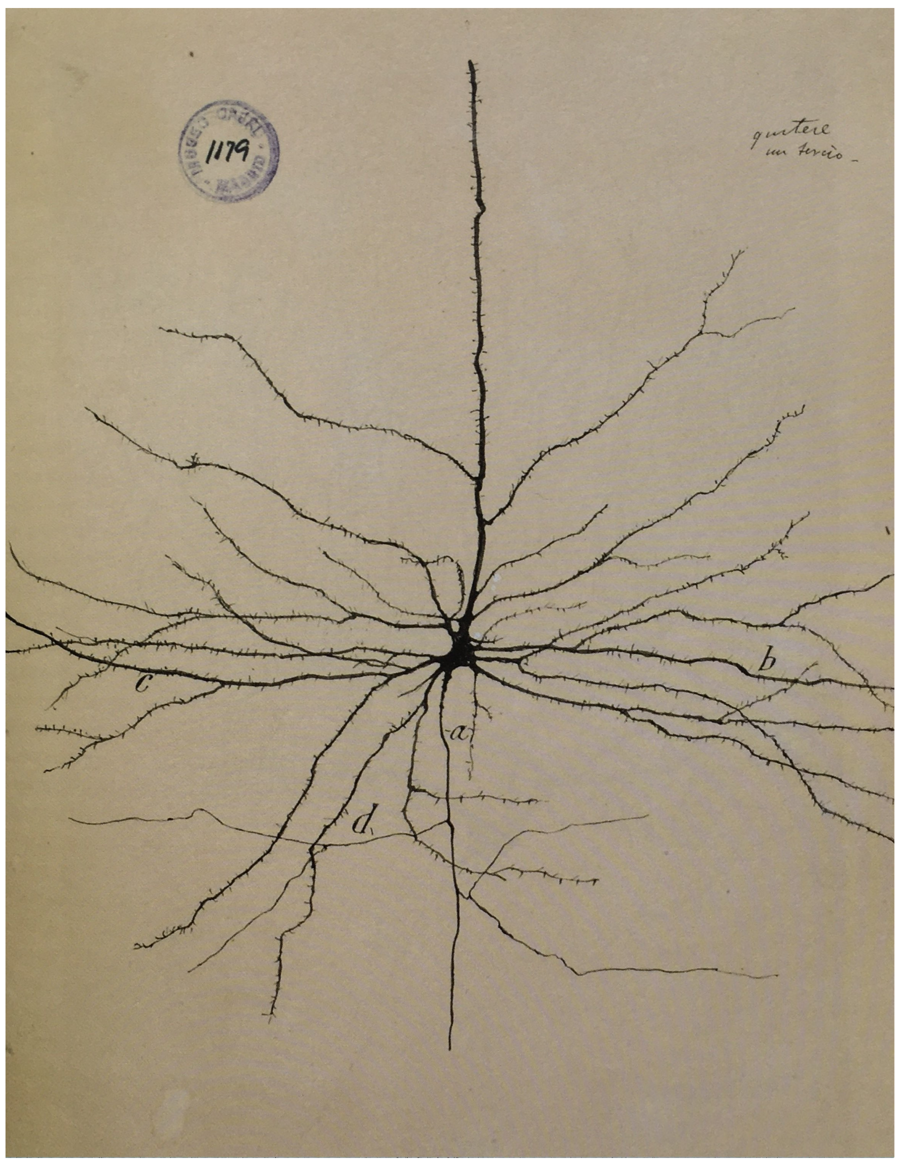


Olfactory bulb, Camillo Golgi, 1875

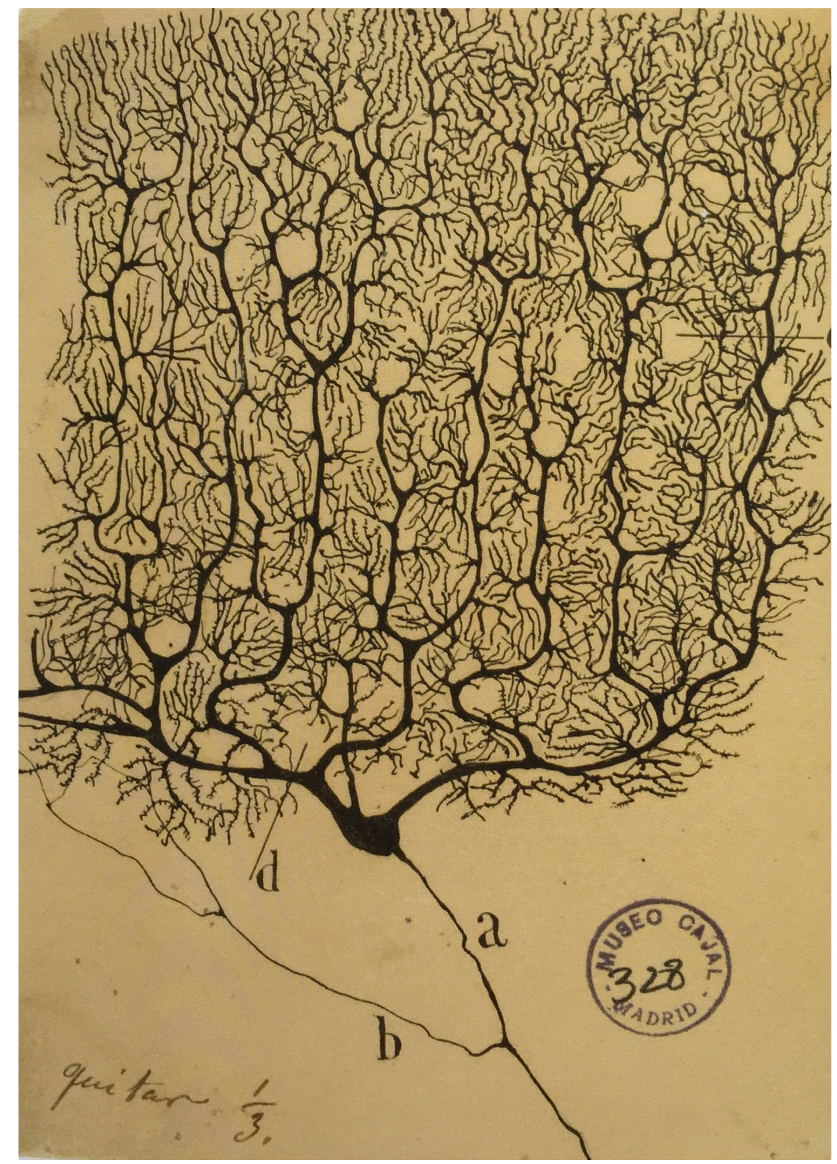
slide by Saeed Saremi

Santiago Ramón Y Cajal & The Neuron Doctrine

- Neuron as the discrete distinct entities in the brain as opposed to a continuous network.



pyramidal neuron. Cajal, 1899



purkinje neuron. Cajal, 1899

slide by Saeed Saremi

Santiago Ramón Y Cajal & The Neuron Doctrine

The Nobel Prize in Physiology or Medicine 1906

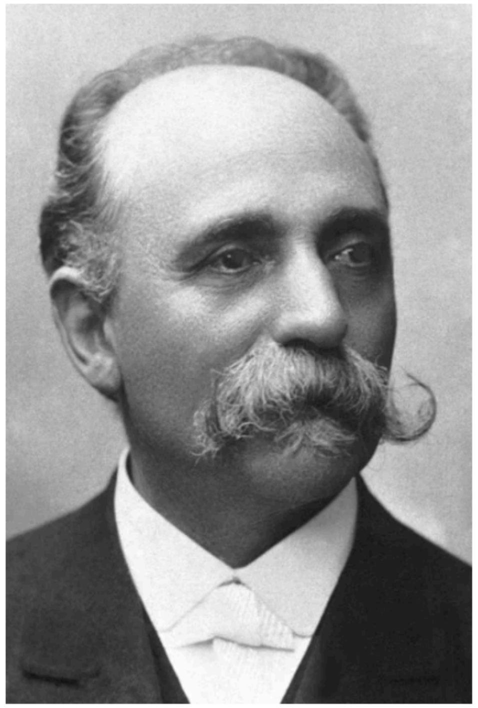


Photo from the Nobel Foundation archive.

Camillo Golgi

Prize share: 1/2

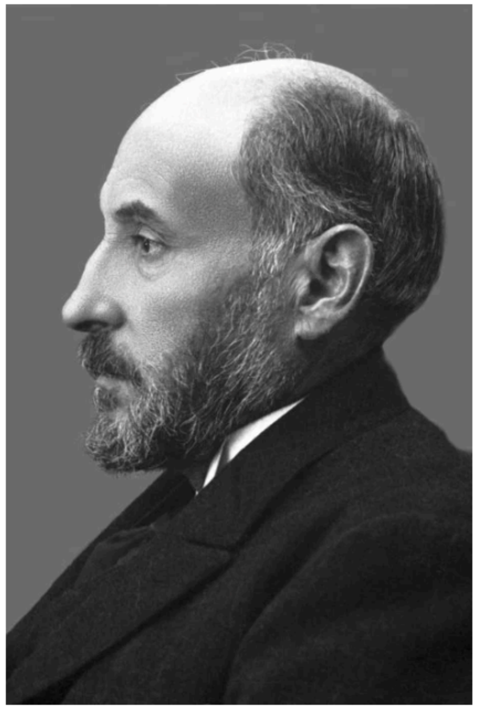


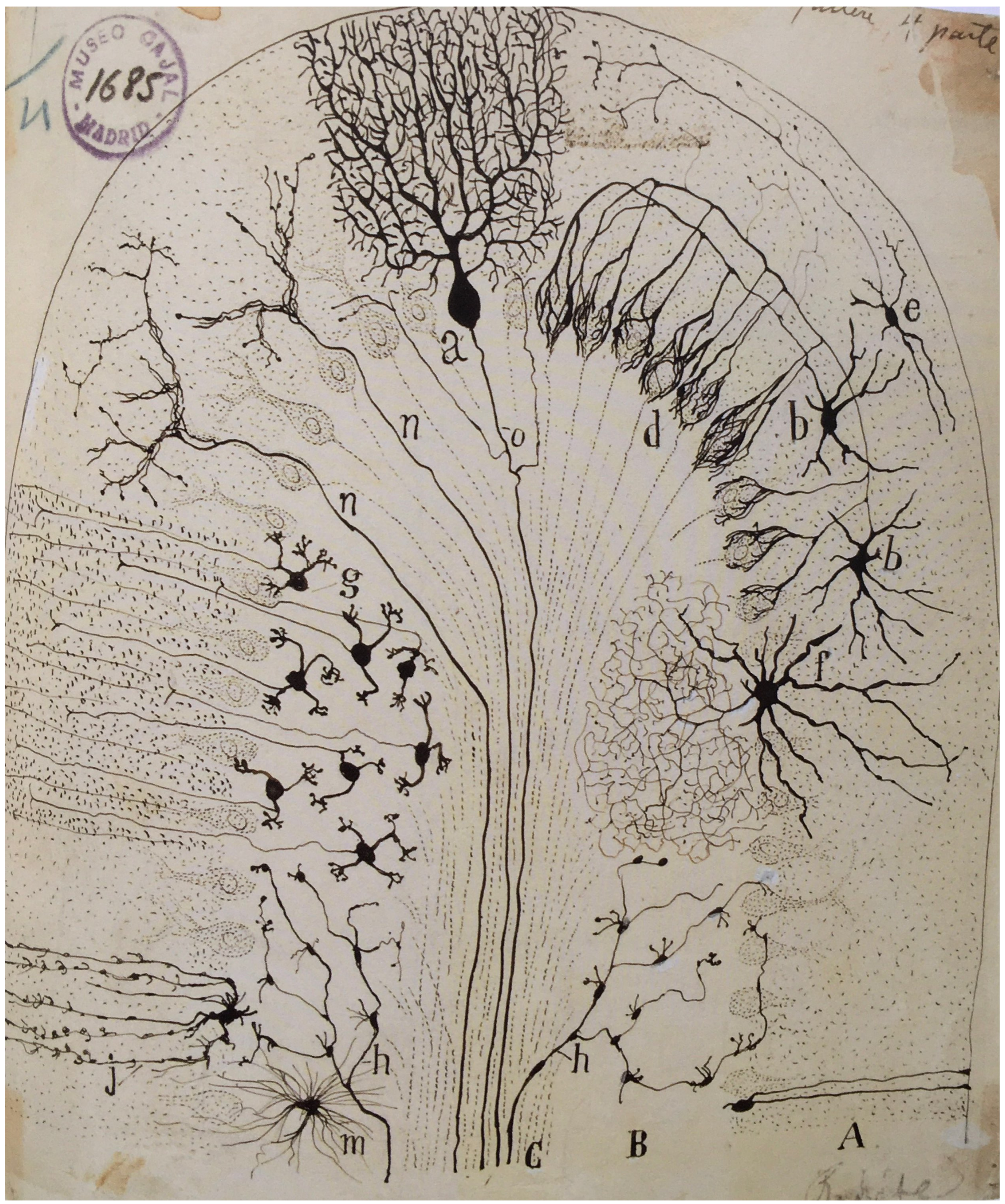
Photo from the Nobel Foundation archive.

Santiago Ramón y Cajal

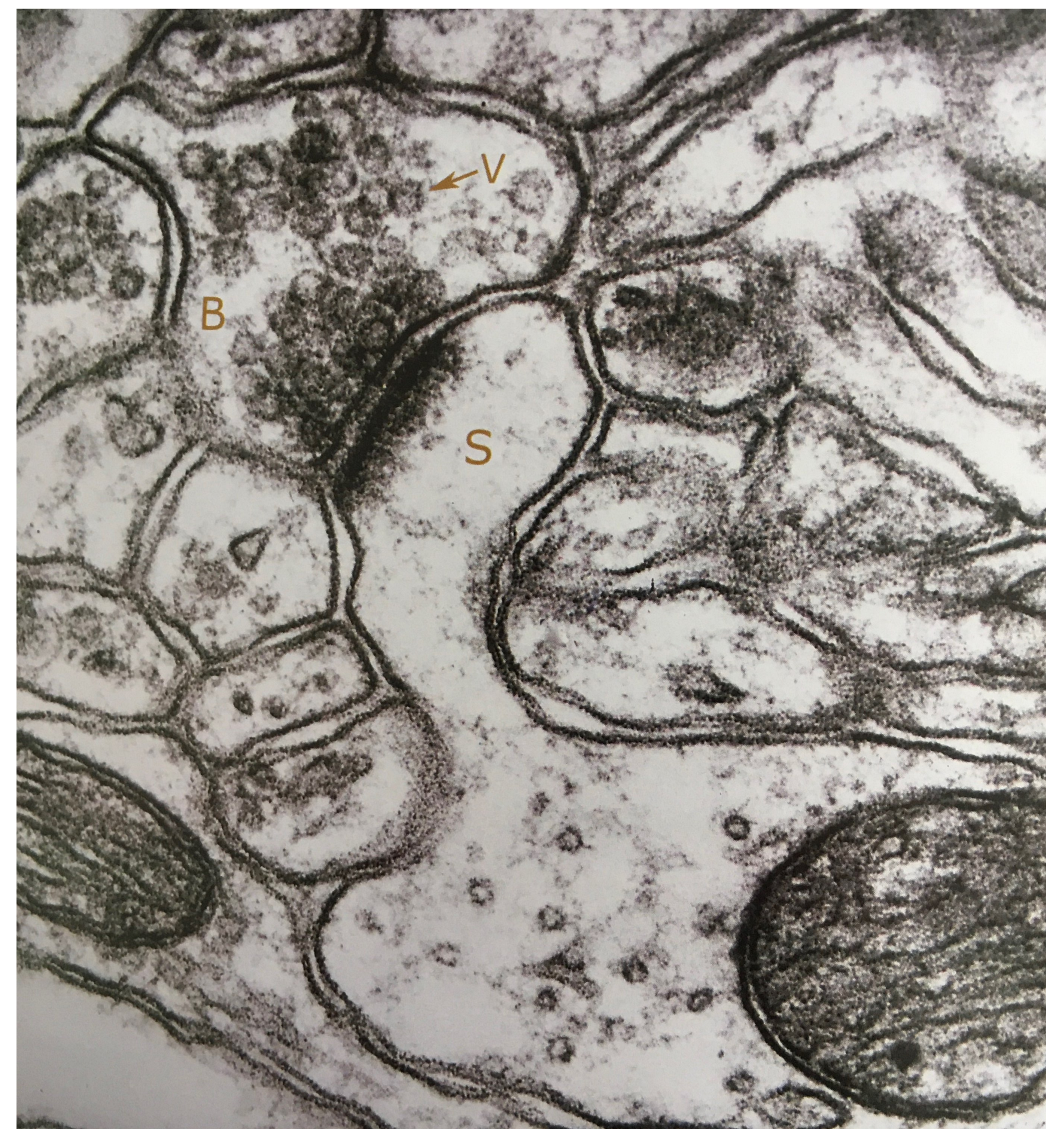
Prize share: 1/2

The Nobel Prize in Physiology or Medicine 1906 was awarded jointly to Camillo Golgi and Santiago Ramón y Cajal "in recognition of their work on the structure of the nervous system"

Network of Neurons:
Axons, Dendrites, and Synapses



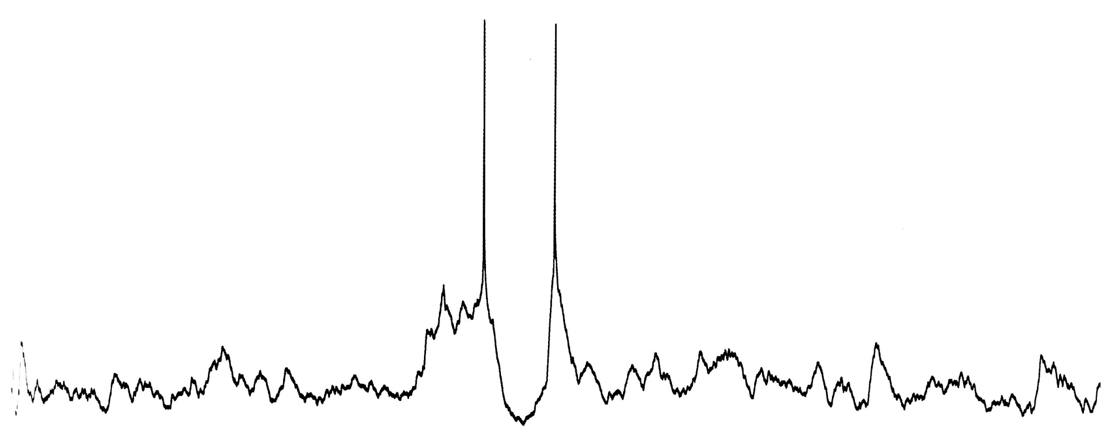
network of neurons. Cajal, 1899



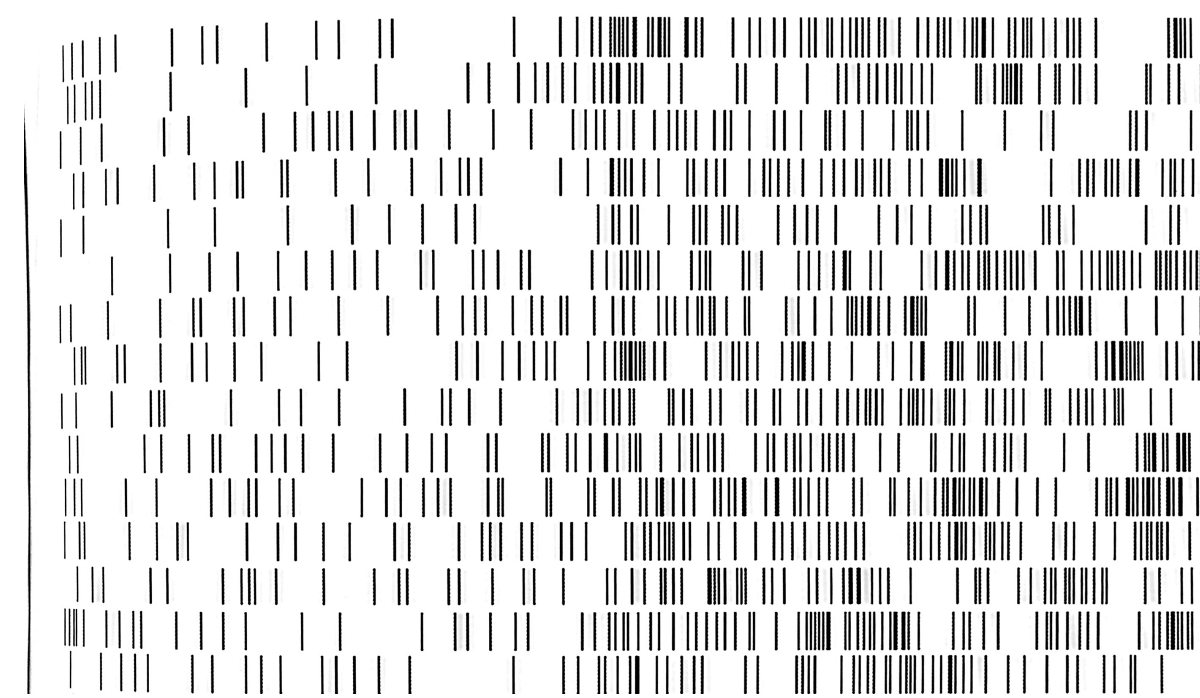
Synapse. Spacek and Harris, 2000

slide by Saeed Saremi

Electricity in the brain



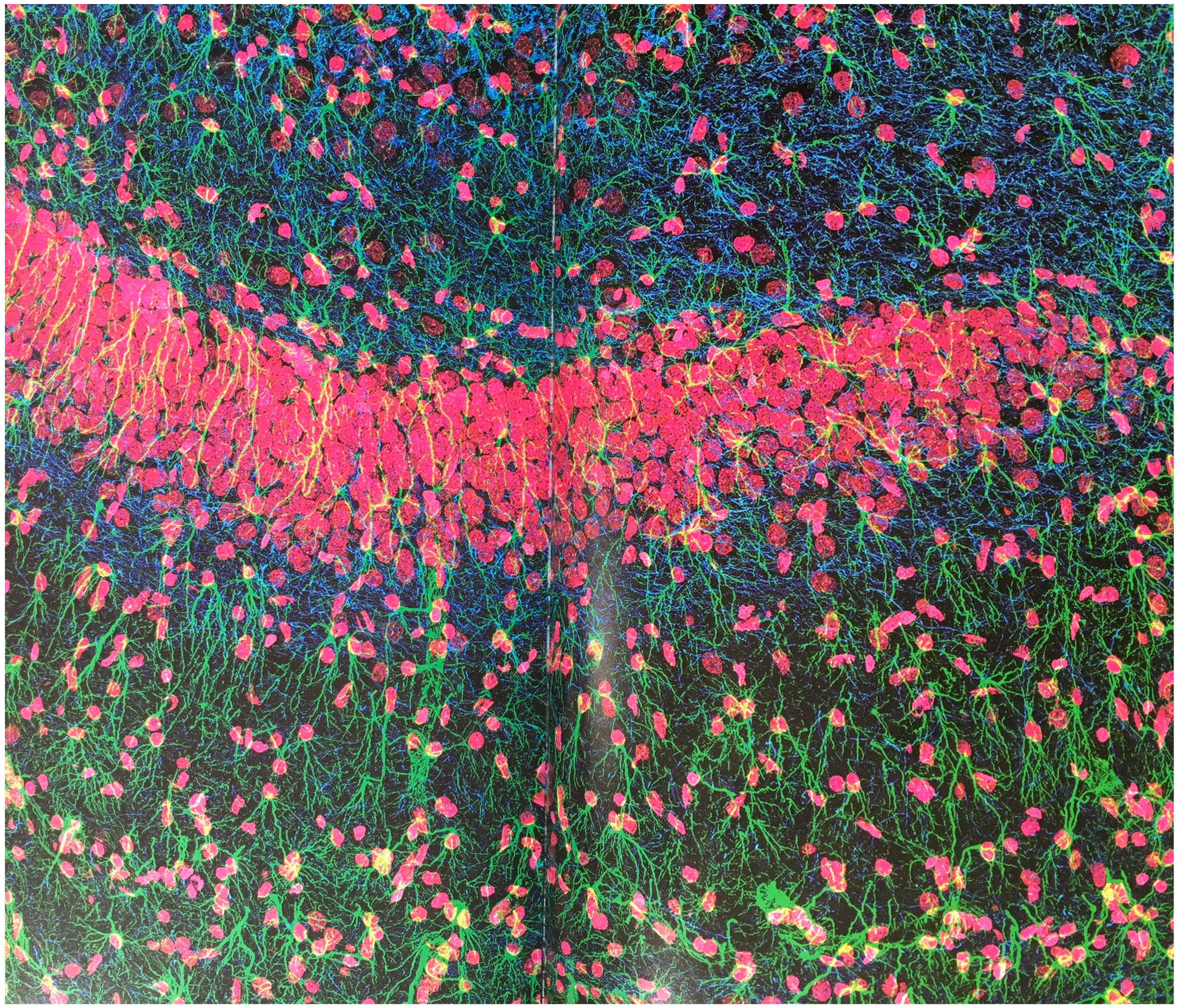
Whole-cell recording in an awake rat. Contantinople and Bruno, 2009



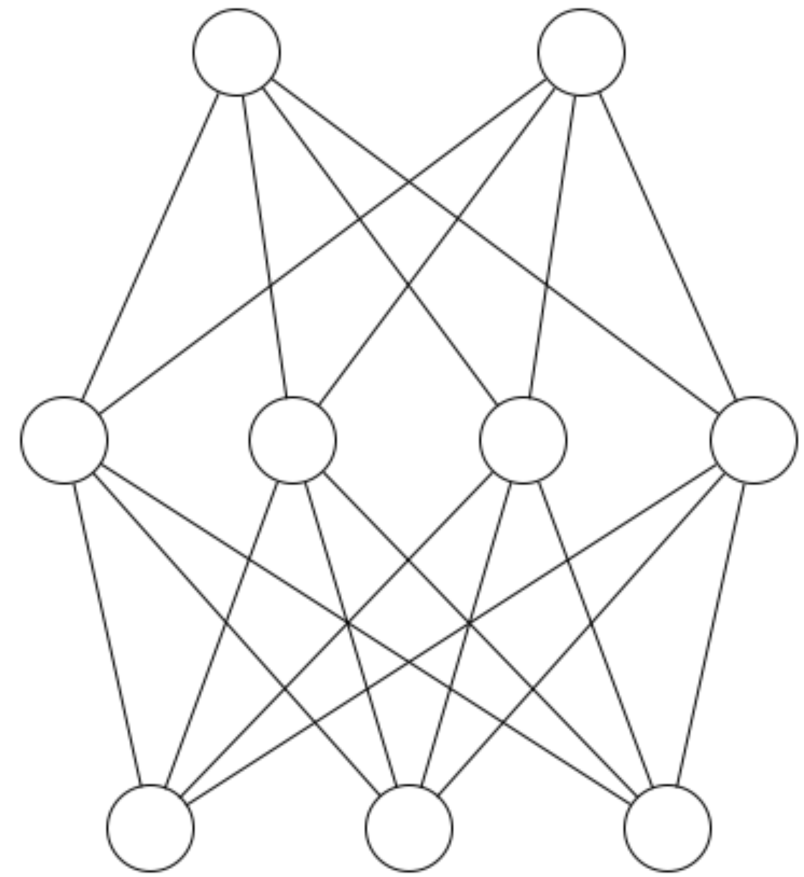
Action potentials in a live monkey brain. Saez and Salzman, 2009. (Each row, 4 seconds.)

slide by Saeed Saremi

Electricity in the brain



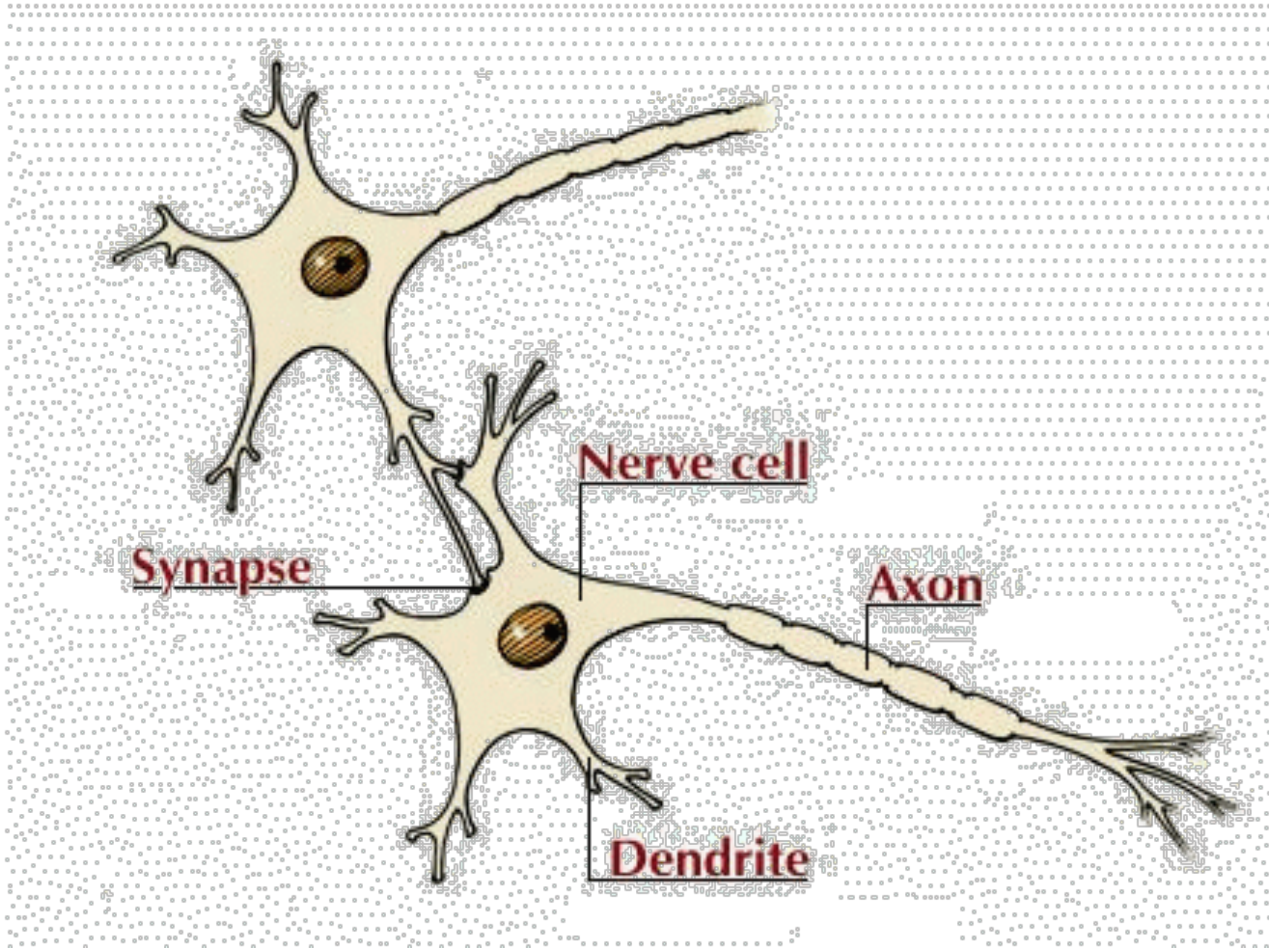
human brain: 10^{11} neurons, 10^{15} synapses



GPT-4: 10^{6} neurons, 10^{11} parameters (weights), 100 layers

slide by Saeed Saremi

Neurons



- Soma (CPU)
  Cell body - combines signals

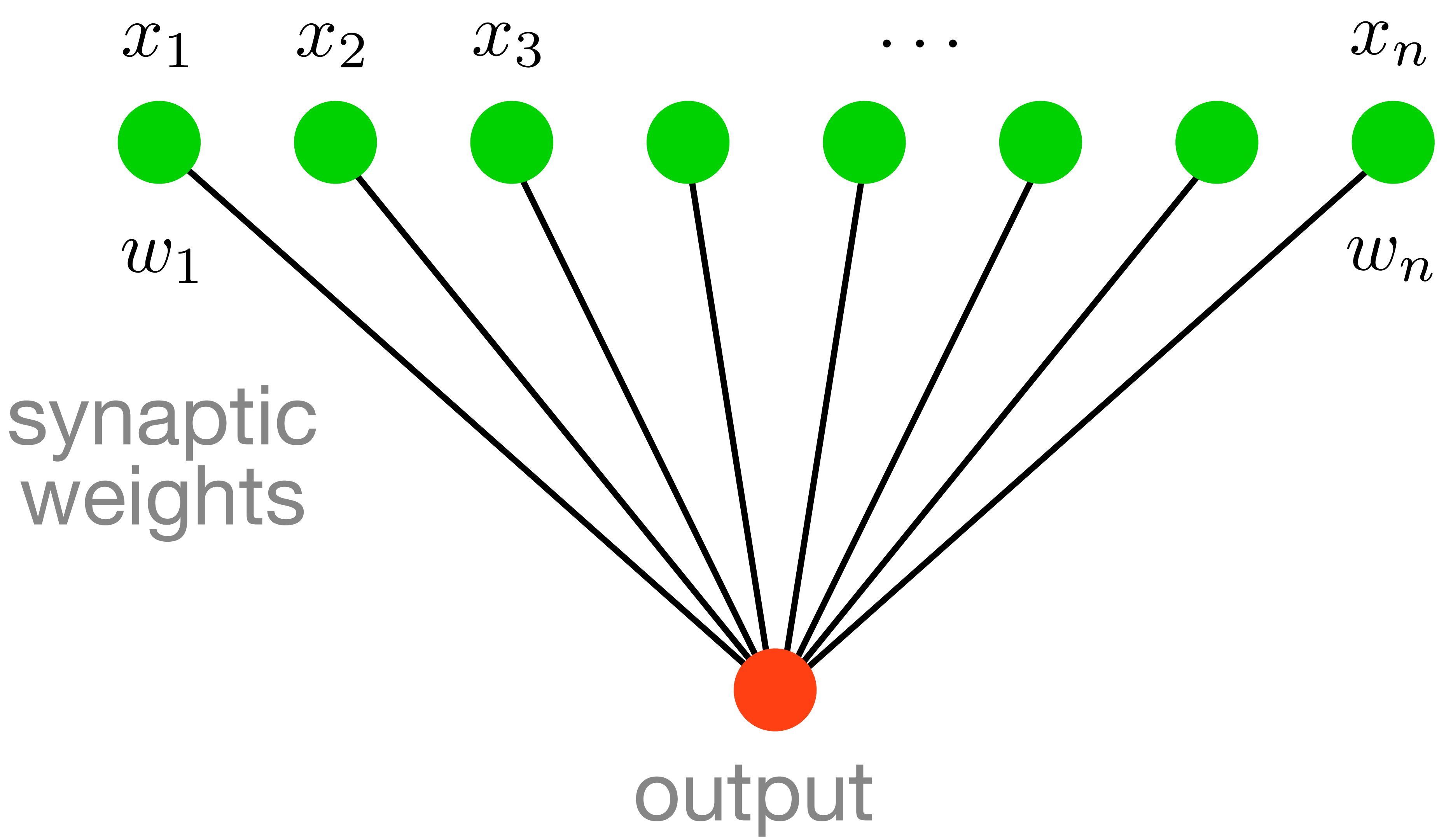
- Dendrite (input bus)
  Combines the inputs from several other nerve cells

- 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

slide by Alex Smola

Neurons

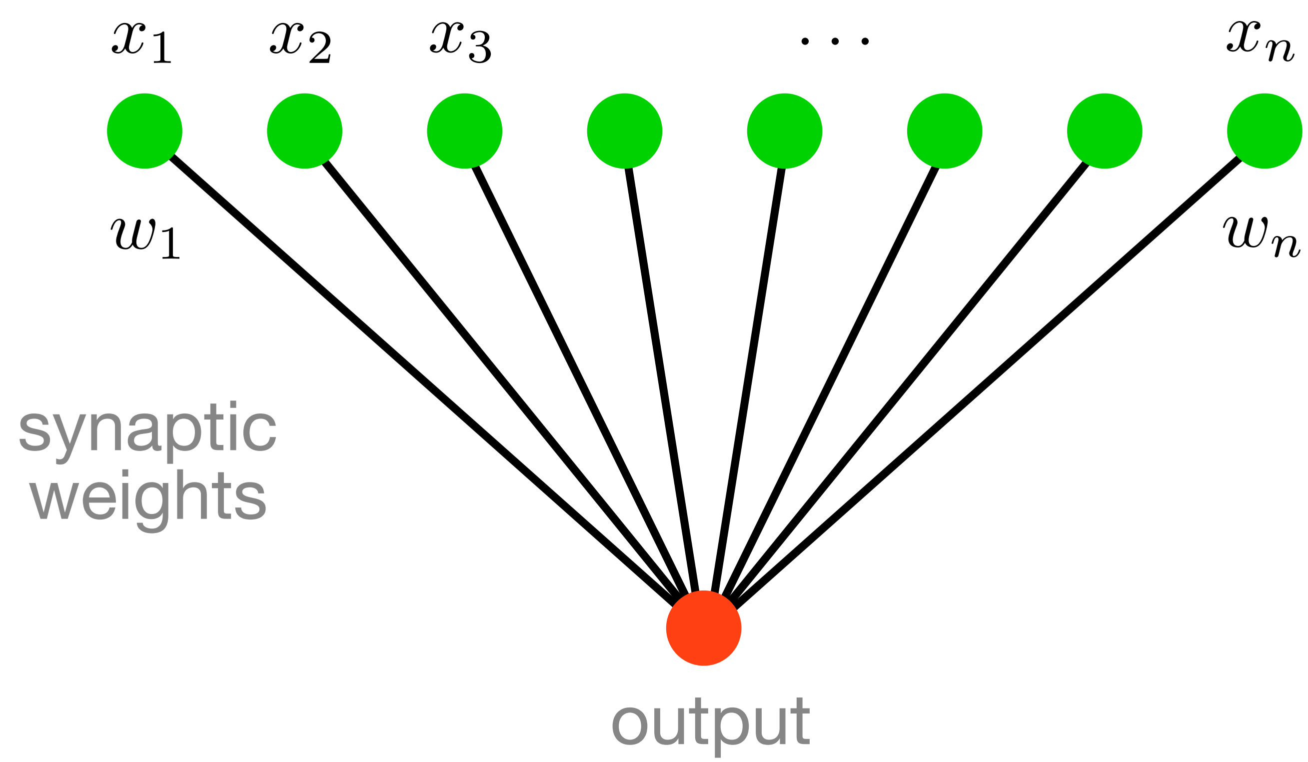


$$f(x) = \sum_i w_i x_i = \langle w, x \rangle$$

slide by Alex Smola

Perceptron

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

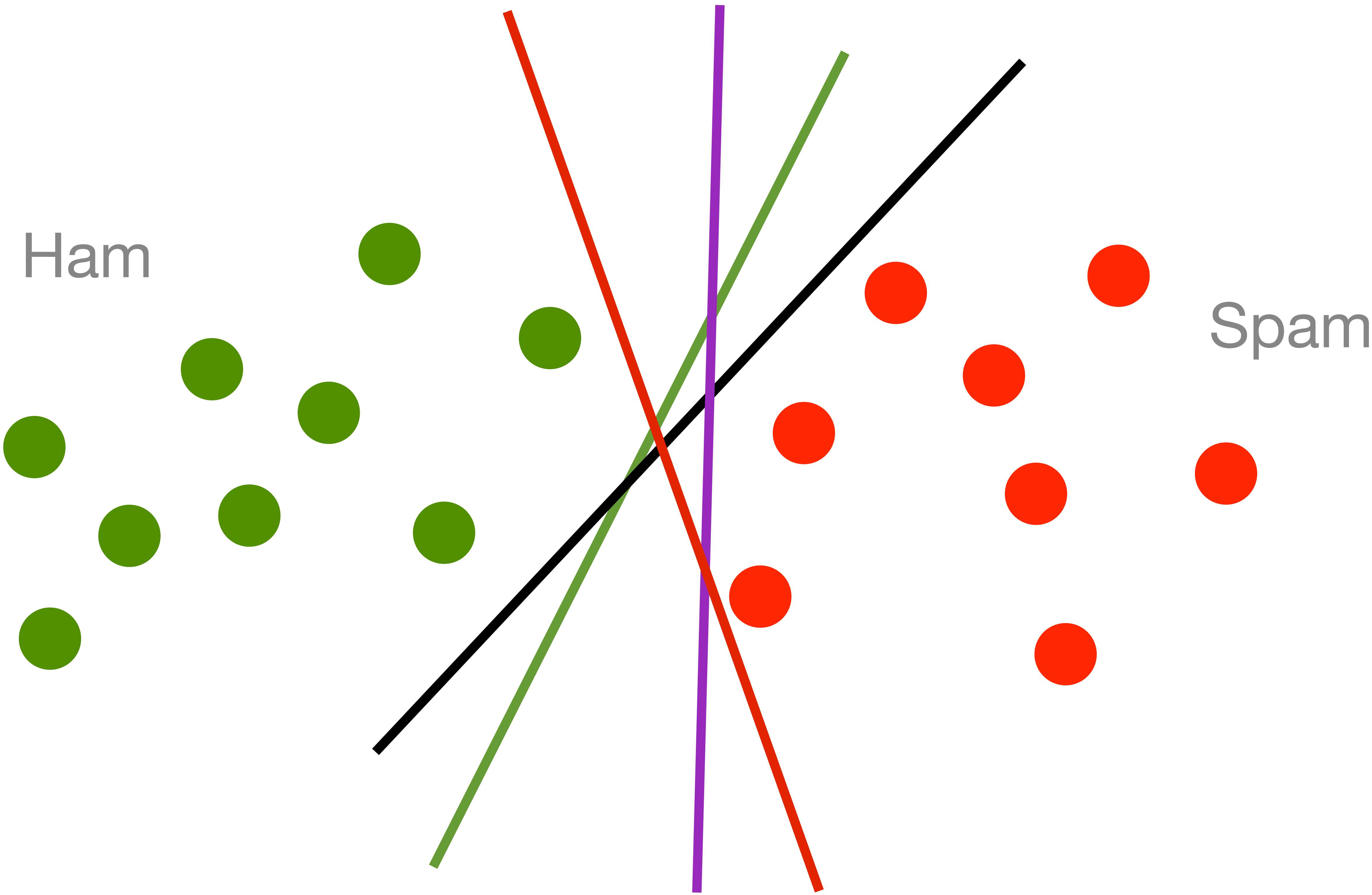


$$f(x) = \sigma\left(\langle w, x\rangle + b\right)$$

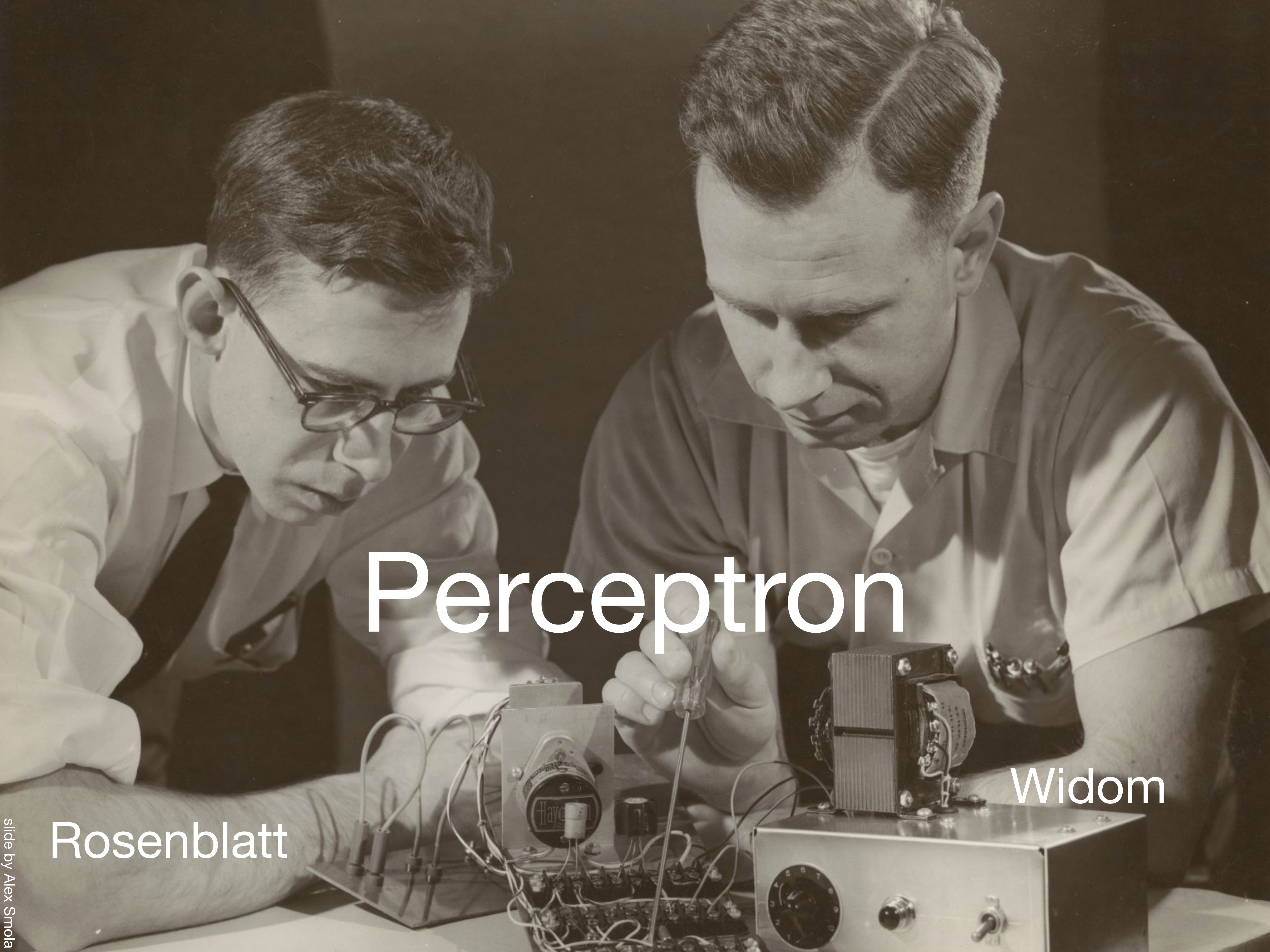
- Linear separating hyperplanes (spam/ham, novel/typical, click/no click)
- Learning
  Estimating the parameters w and b

slide by Alex Smola

Perceptron



slide by Alex Smola


Perceptron
Widom
Rosenblatt
slide by Alex Smola

Perceptron Inductive Bias

1. Decision boundary should be linear
2. Most recent mistakes are most important (and should be corrected)

Background: Hyperplanes

Notation Trick: fold the bias *b* and the weights ***w*** into a single vector **θ** by prepending a constant to ***x*** and increasing dimensionality by one to get **x'**!

Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} = b\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x'} : \boldsymbol{\theta}^T\mathbf{x'} = 0 \text{ and } x'_1 = 1\}$$

$$\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$$

Half-spaces:

$$\mathcal{H}^+ = \{\mathbf{x} : \boldsymbol{\theta}^T\mathbf{x} > 0 \text{ and } x_1 = 1\}$$

$$\mathcal{H}^- = \{\mathbf{x} : \boldsymbol{\theta}^T\mathbf{x} < 0 \text{ and } x_1 = 1\}$$

(Online) Perceptron Algorithm

Data: Inputs are continuous vectors of length *M*. Outputs are discrete.

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

Prediction: Output determined by hyperplane.

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

$$\text{sign}(a) = \begin{cases} 1, & \text{if } a \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$ and $x_1 = 1$

Learning: Iterative procedure:

- **initialize** parameters to vector of all zeroes
- **while** not converged
  - **receive** next example ($\mathbf{x}^{(i)}$, $y^{(i)}$)
  - **predict** $y' = h(\mathbf{x}^{(i)})$
  - **if** positive mistake: **add** $\mathbf{x}^{(i)}$ to parameters
  - **if** negative mistake: **subtract** $\mathbf{x}^{(i)}$ from parameters

slide by Matt Gormley & Henry Chai

(Online) Perceptron Algorithm

Data: Inputs are continuous vectors of length *M*. Outputs are discrete.

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

Prediction: Output determined by hyperplane.

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

$$\text{sign}(a) = \begin{cases} 1, & \text{if } a \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$ and $x_1 = 1$

Learning:

Algorithm 1 Perceptron Learning Algorithm (Online)

```
procedure PERCEPTRON(𝒟 = {(x^(1), y^(1)), (x^(2), y^(2)), ...})
    θ ← 0                                 ▷ Initialize parameters
    for i ∈ {1, 2, ...} do                ▷ For each example
        ŷ ← sign(θ^T x^(i))               ▷ Predict
        if ŷ ≠ y^(i) then                 ▷ If mistake
            θ ← θ + y^(i) x^(i)           ▷ Update parameters
    return θ
```

(Online) Perceptron Algorithm

Data: Inputs are continuous vectors of length *M*. Outputs are discrete.

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

Prediction: Output determine

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]$

Learning:

Algorithm 1 Perceptron Learning Alg

procedure PERCEPTRON($\mathcal{D} = \{(\mathbf{x}$

$\boldsymbol{\theta} \leftarrow \mathbf{0}$ ▷ Initialize parameters

for $i \in \{1, 2, \ldots\}$ **do** ▷ For each example

$\hat{y} \leftarrow$ sign$(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$ ▷ Predict

if $\hat{y} \neq y^{(i)}$ **then** ▷ If mistake

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \mathbf{x}^{(i)}$ ▷ Update parameters

return $\boldsymbol{\theta}$

Implementation Trick: same behavior as our "*add on positive mistake and subtract on negative mistake*" version, because $y^{(i)}$ takes care of the sign

slide by Matt Gormley & Henry Chai

(Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset, D. We call this the "batch" setting in contrast to the "online" setting that we've discussed so far.

Algorithm 1 Perceptron Learning Algorithm (Batch)

```
procedure PERCEPTRON(𝒟 = {(x^(1), y^(1)), ..., (x^(N), y^(N))})
    θ ← 0                                   ▷ Initialize parameters
    while not converged do
        for i ∈ {1, 2, ..., N} do            ▷ For each example
            ŷ ← sign(θ^T x^(i))              ▷ Predict
            if ŷ ≠ y^(i) then                ▷ If mistake
                θ ← θ + y^(i) x^(i)          ▷ Update parameters
    return θ
```

$$\textbf{procedure } \textsc{Perceptron}(\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\})$$
$$\boldsymbol{\theta} \leftarrow \mathbf{0} \quad \triangleright \text{ Initialize parameters}$$
$$\textbf{while } \text{not converged } \textbf{do}$$
$$\textbf{for } i \in \{1, 2, \ldots, N\} \textbf{ do} \quad \triangleright \text{ For each example}$$
$$\hat{y} \leftarrow \text{sign}(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) \quad \triangleright \text{ Predict}$$
$$\textbf{if } \hat{y} \neq y^{(i)} \textbf{ then} \quad \triangleright \text{ If mistake}$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \mathbf{x}^{(i)} \quad \triangleright \text{ Update parameters}$$
$$\textbf{return } \boldsymbol{\theta}$$

(Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset, D. We call this the "batch" setting in contrast to the "online" setting that we've discussed so far.

Discussion:

The Batch Perceptron Algorithm can be derived in two ways.

1. By extending the online Perceptron algorithm to the batch setting (as mentioned above)
2. By applying **Stochastic Gradient Descent (SGD)** to minimize a so-called **Hinge Loss** on a linear separator

The Perceptron

initialize $w = 0$ and $b = 0$
repeat
  if $y_i \left[\langle w, x_i \rangle + b\right] \leq 0$ **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 $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$

slide by Alex Smola

Convergence Theorem

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

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

slide by Alex Smola

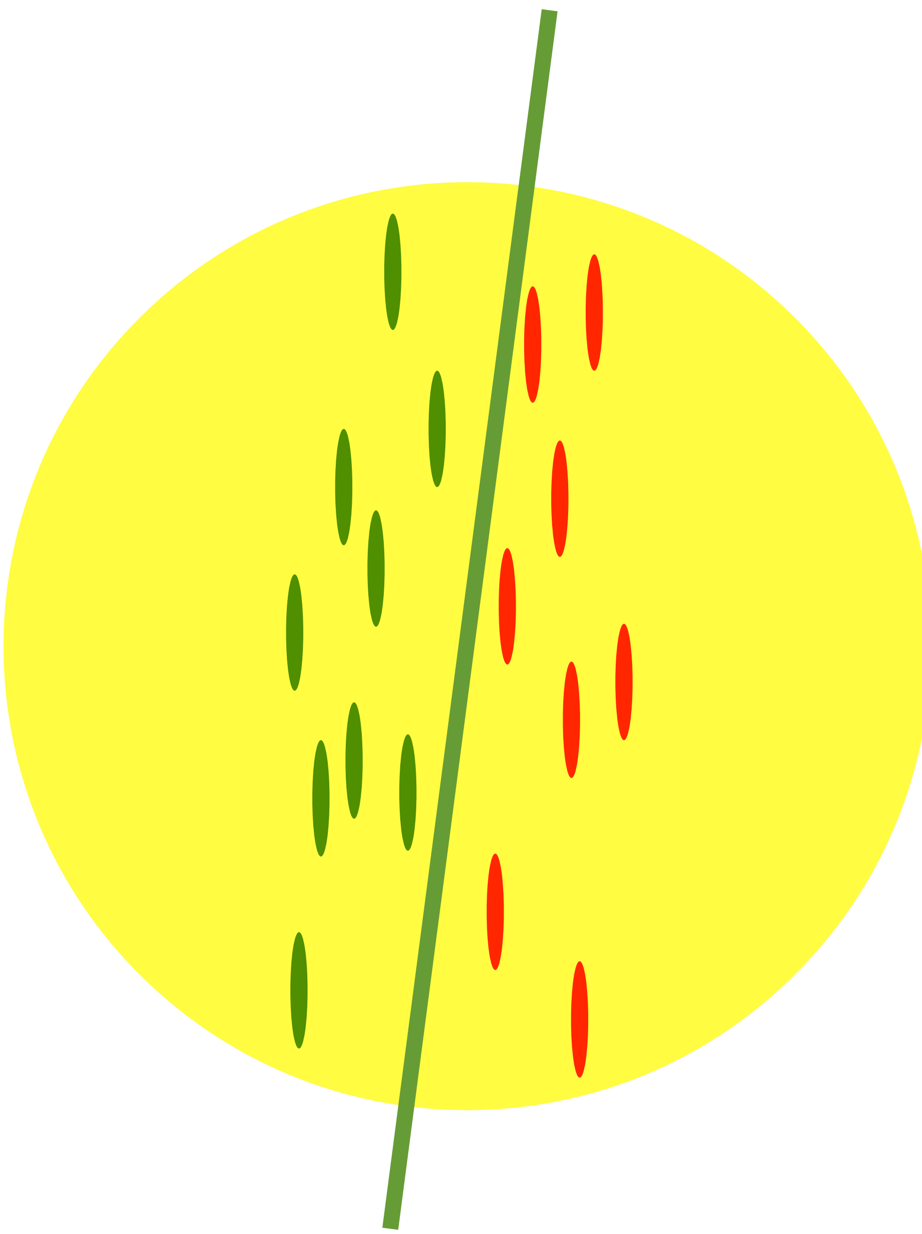
Consequences

- Only need to store errors.
  This gives a compression bound for perceptron.
- Stochastic gradient descent on hinge loss
  $$l(x_i, y_i, w, b) = \max\left(0, 1 - y_i\left[\langle w, x_i\rangle + b\right]\right)$$
- Fails with noisy data

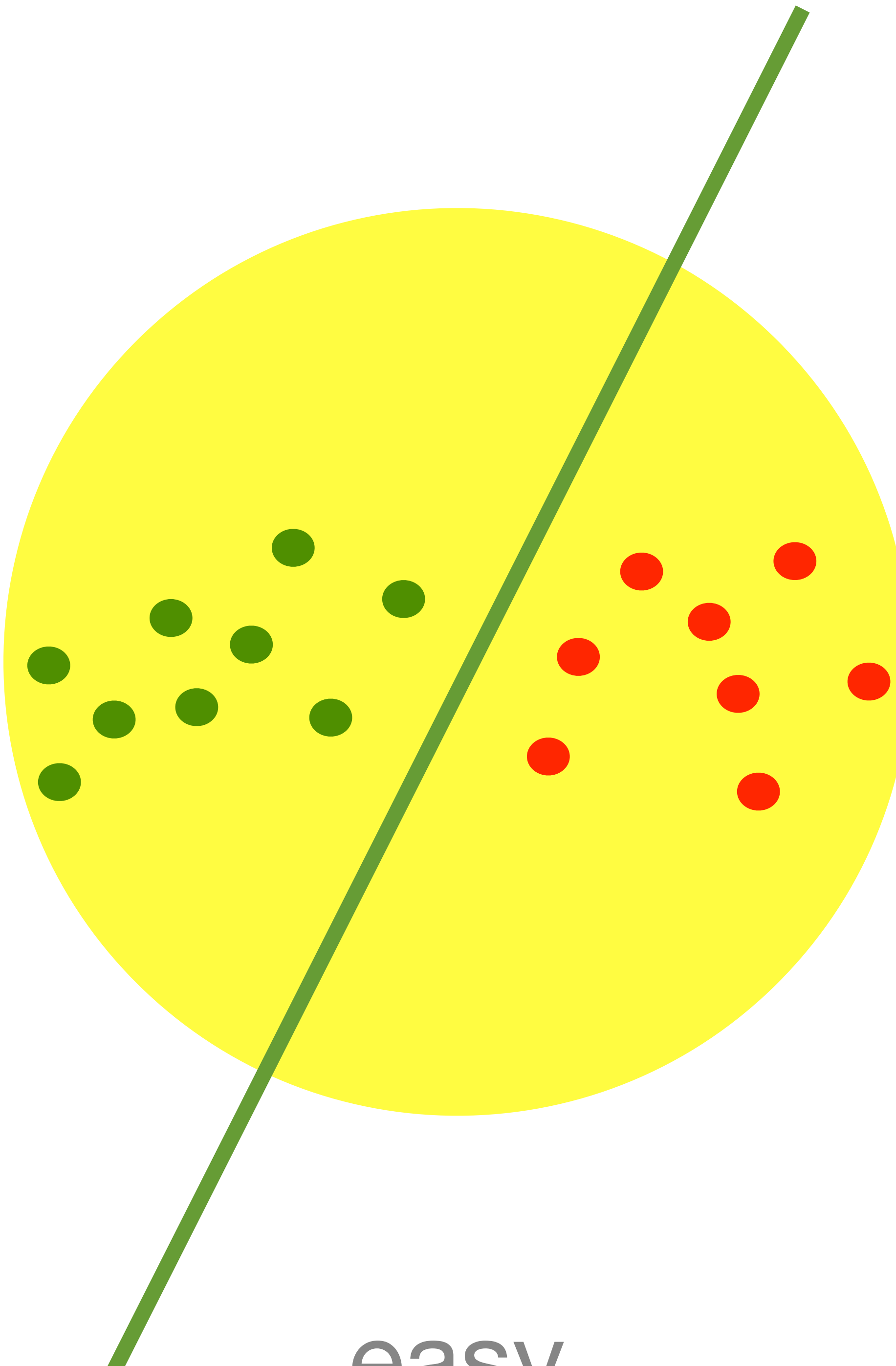
do NOT train your
avatar with perceptrons

slide by Alex Smola

Hardness: margin vs. size

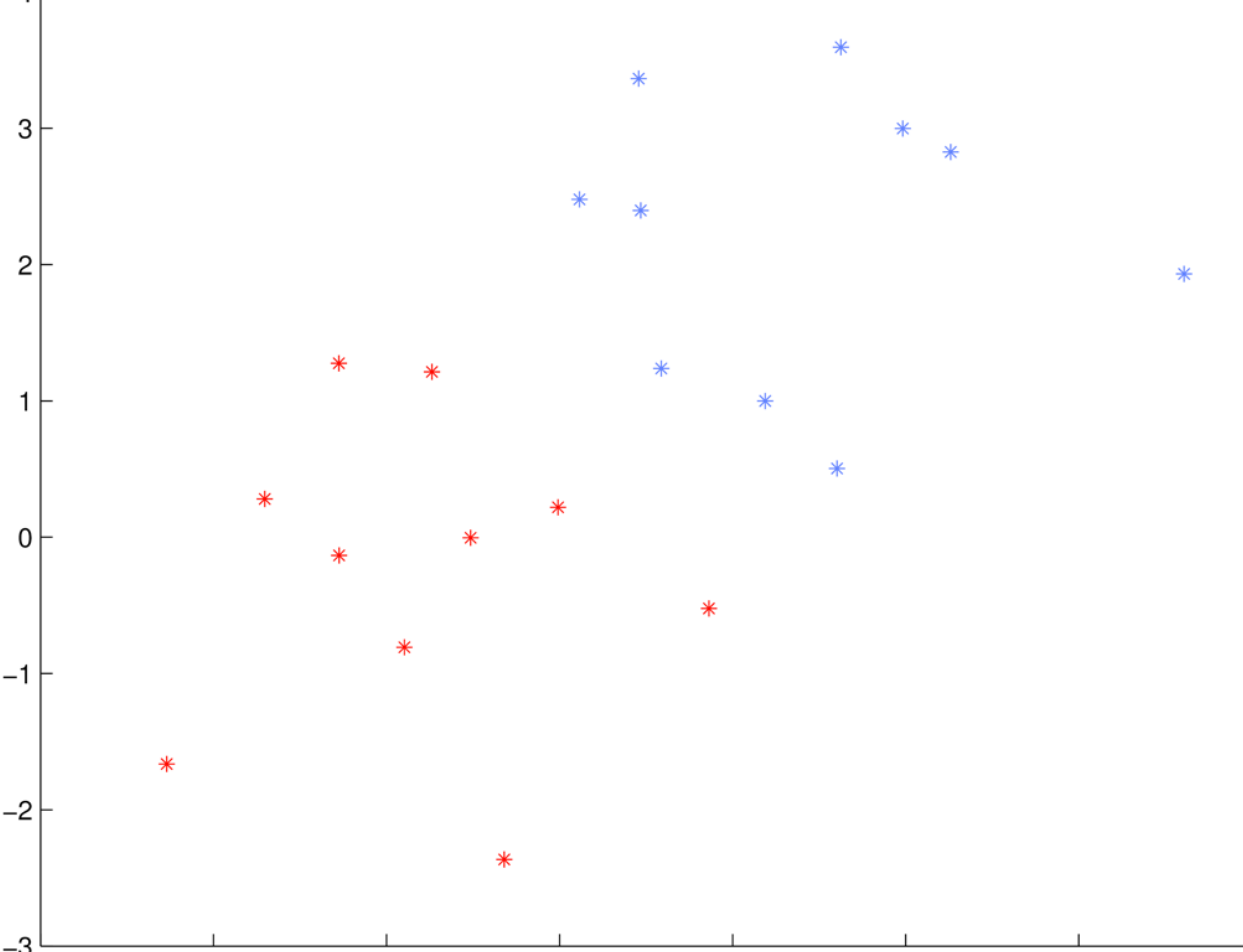


hard

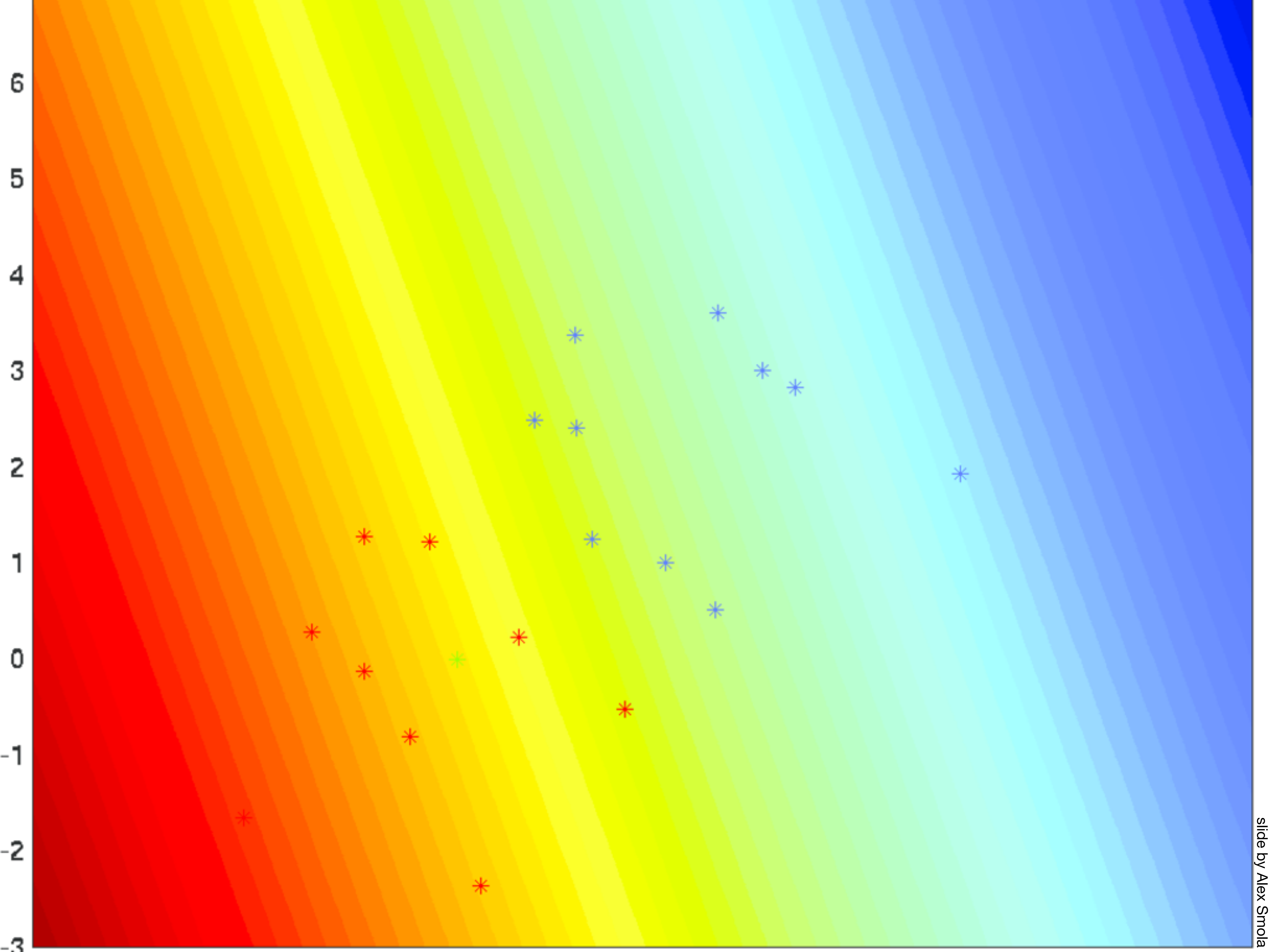


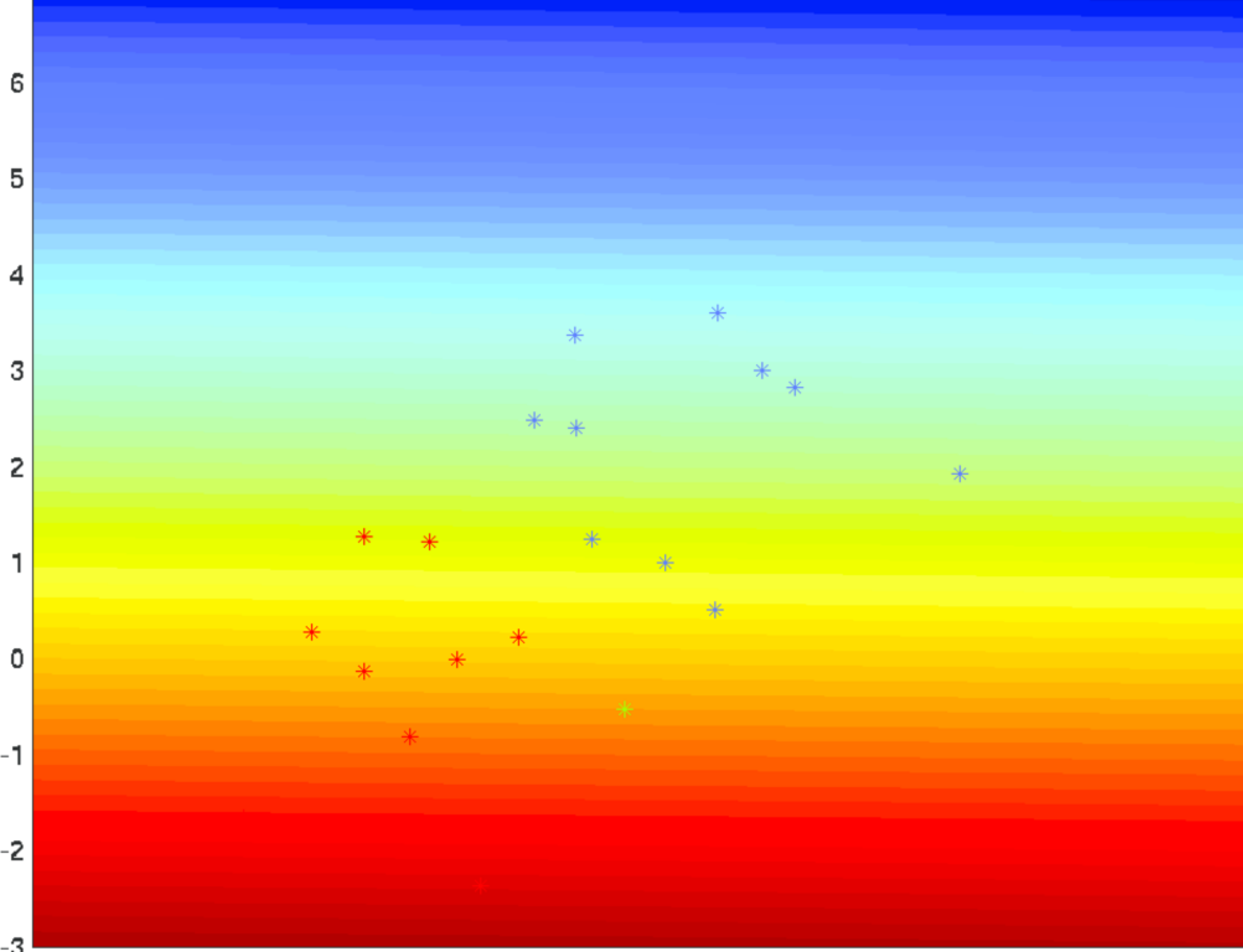
easy

slide by Alex Smola

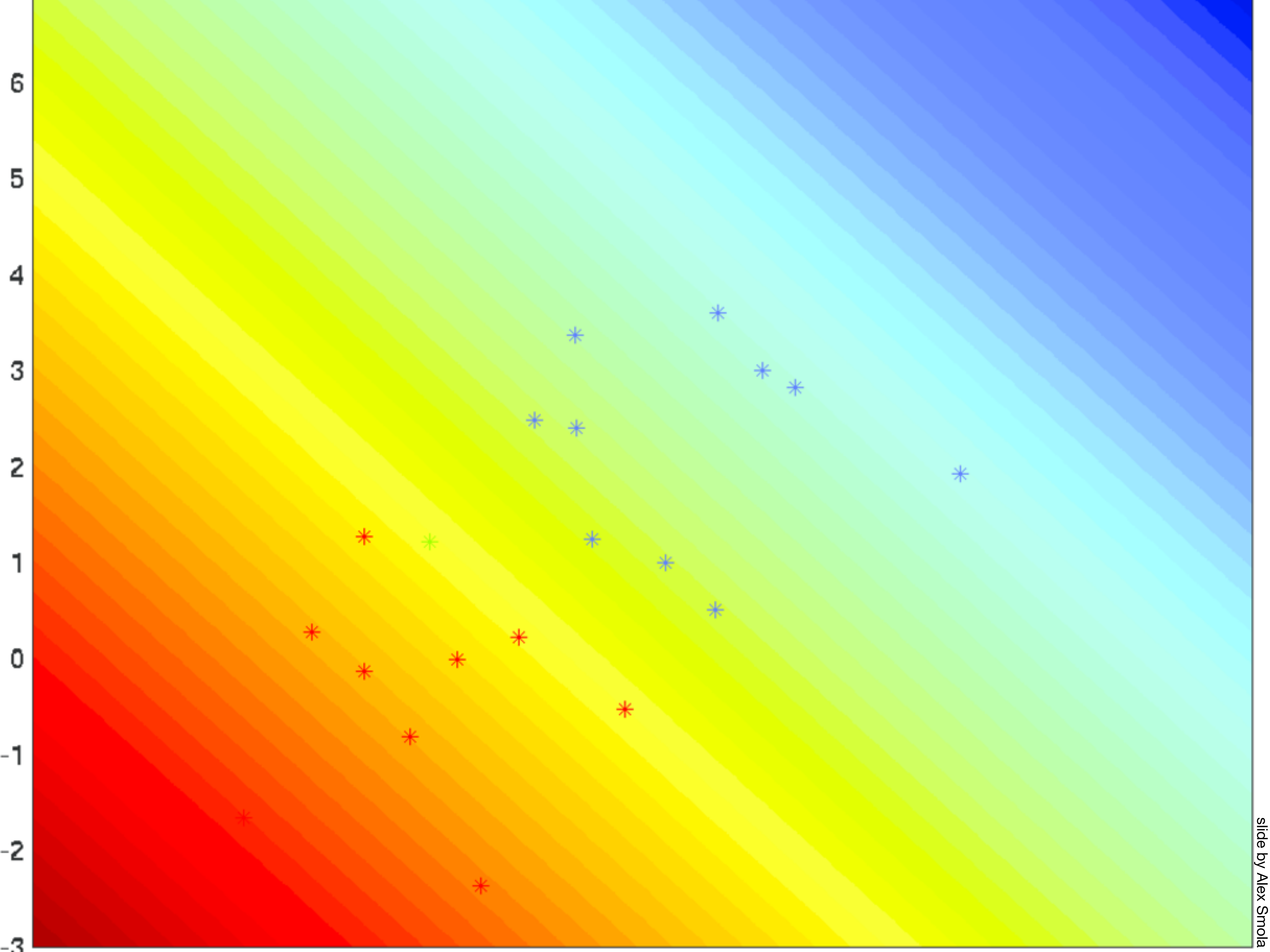


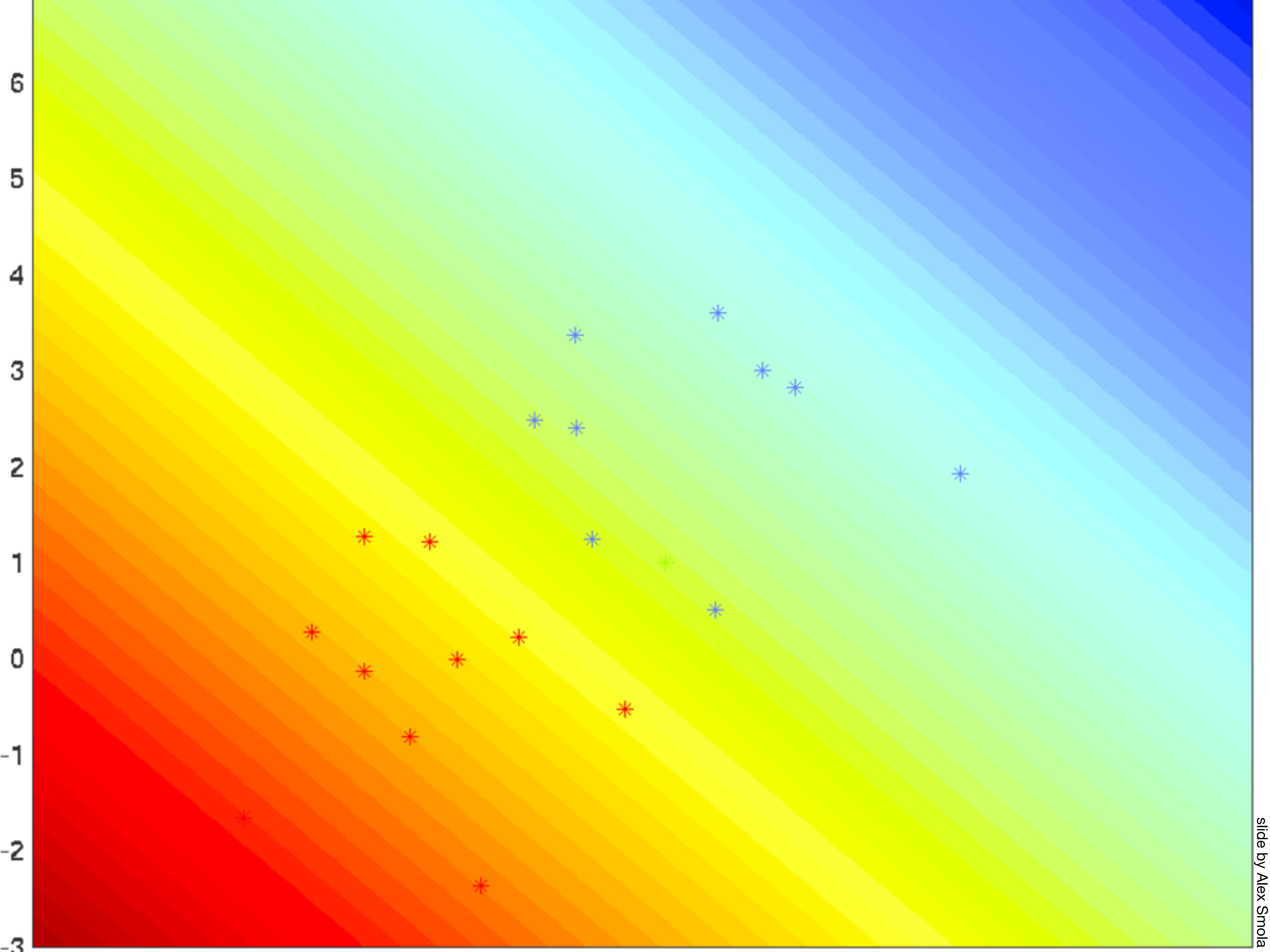
slide by Alex Smola

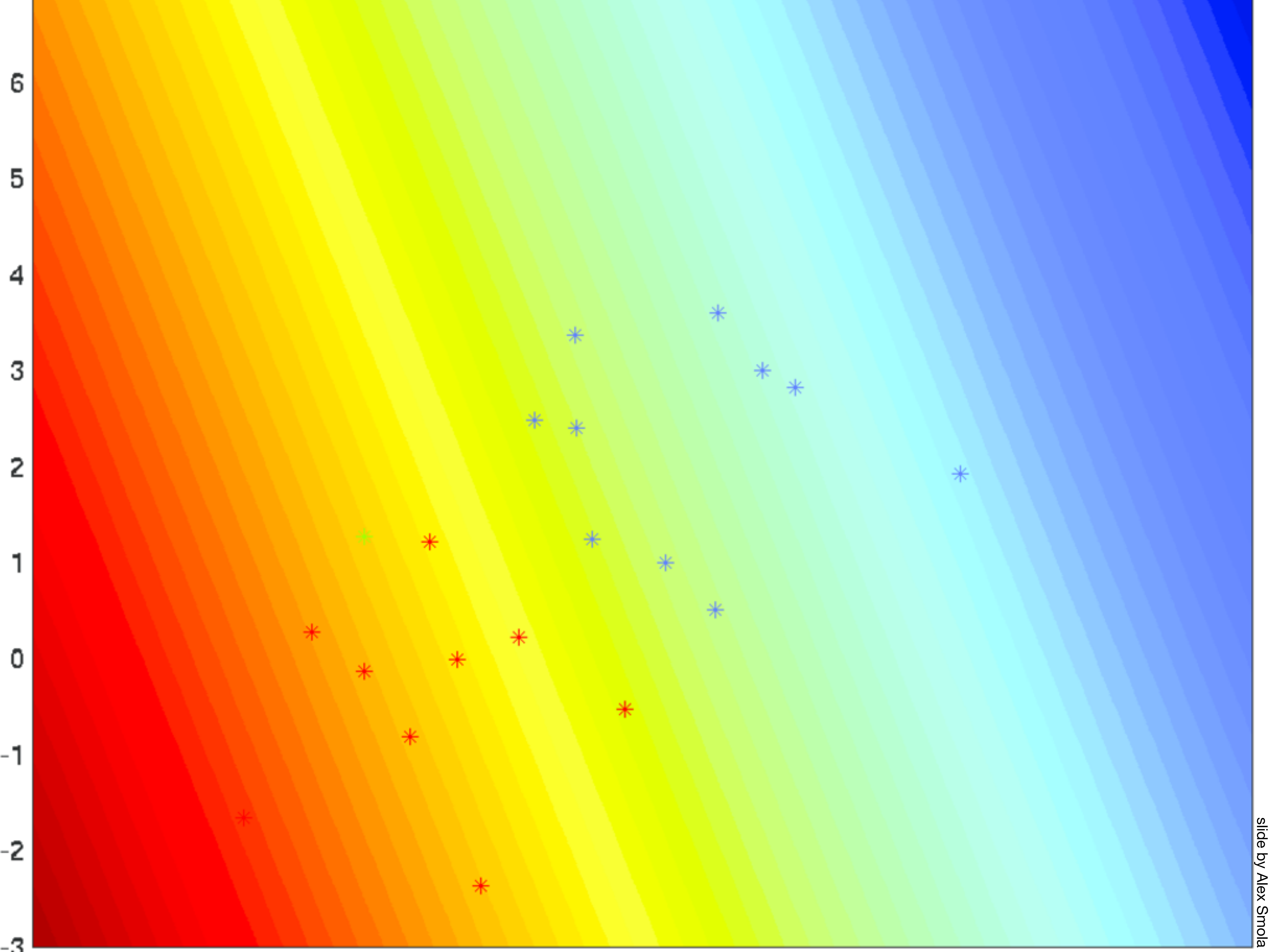

6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola

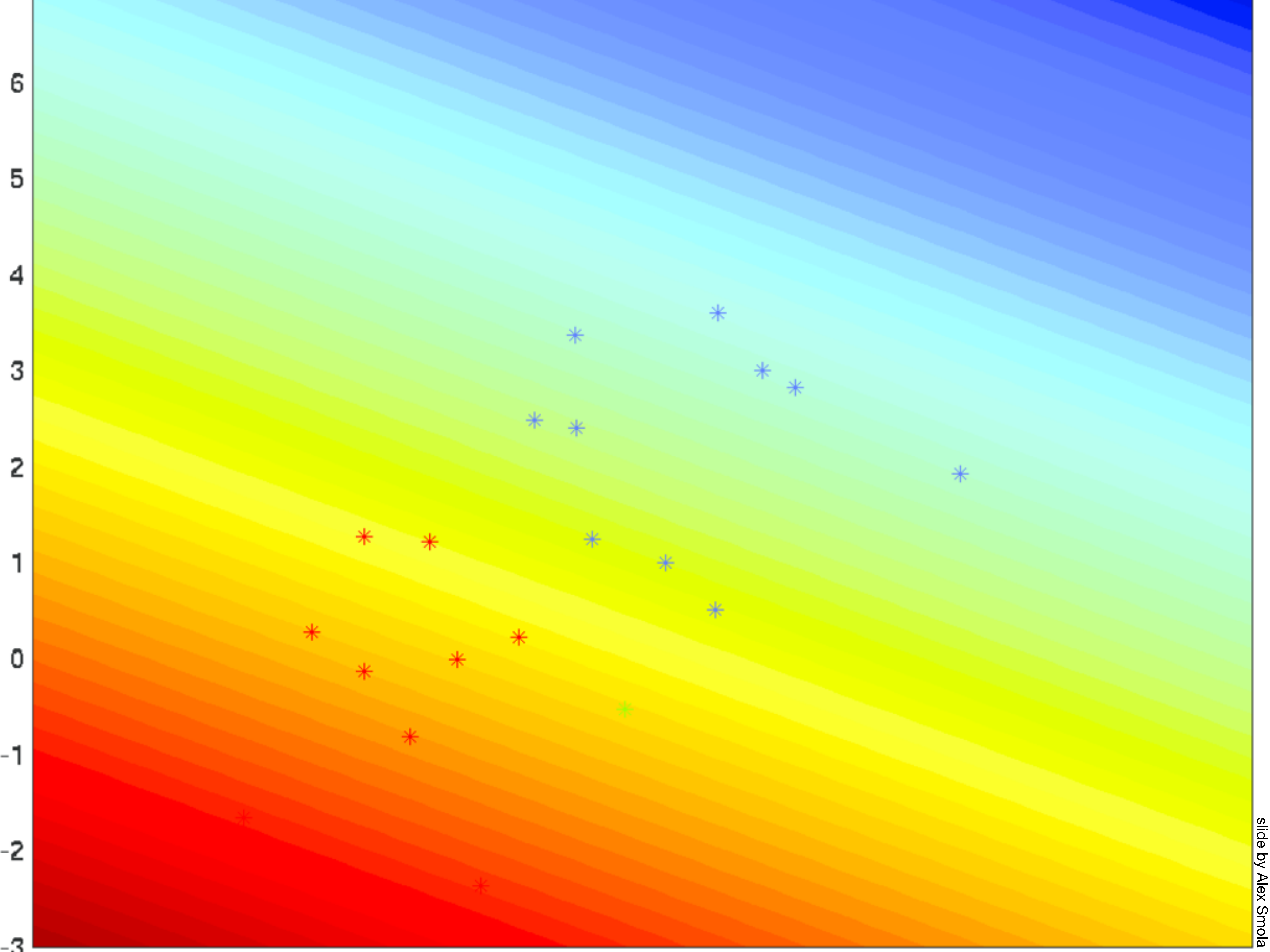


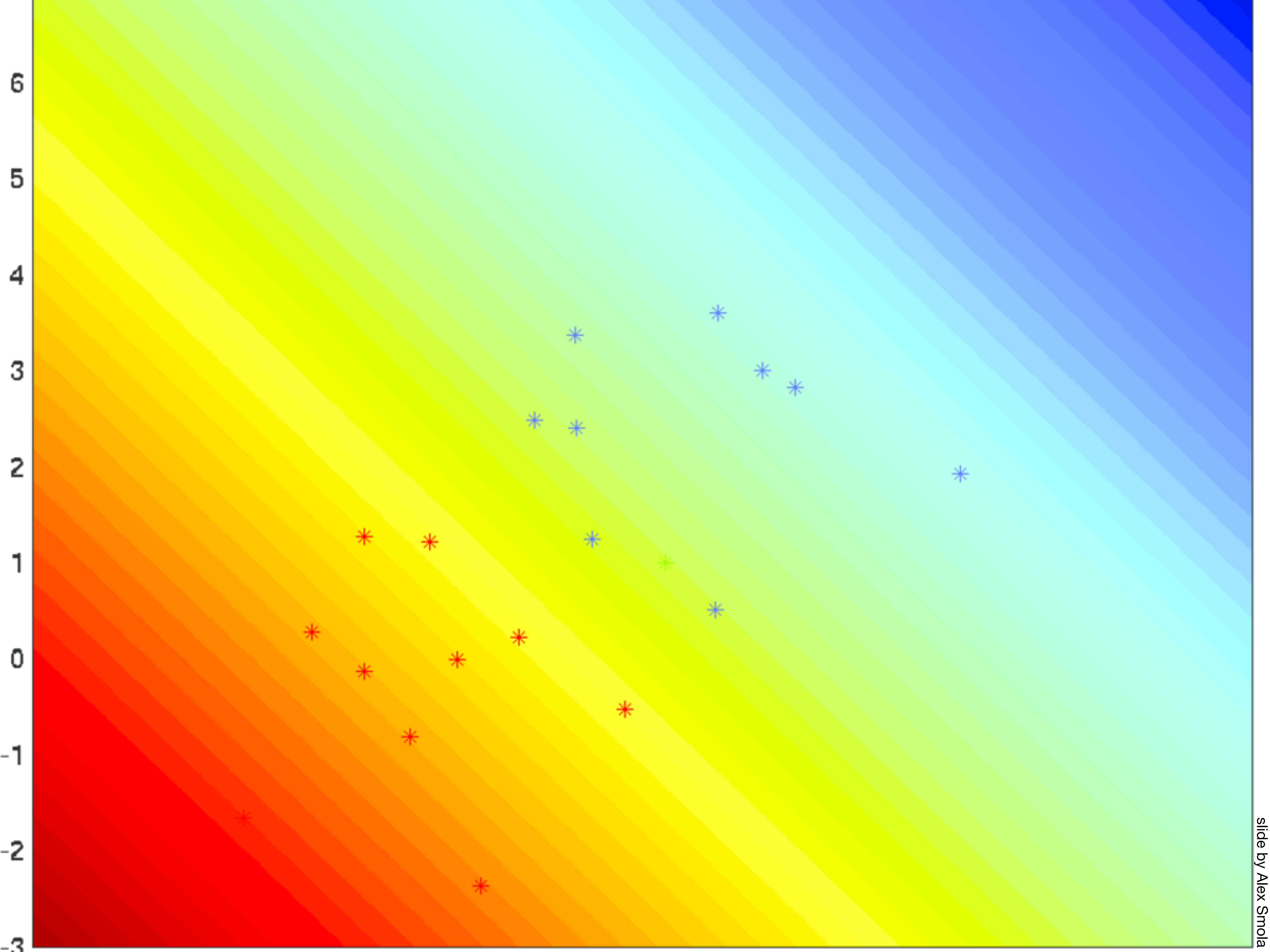
slide by Alex Smola

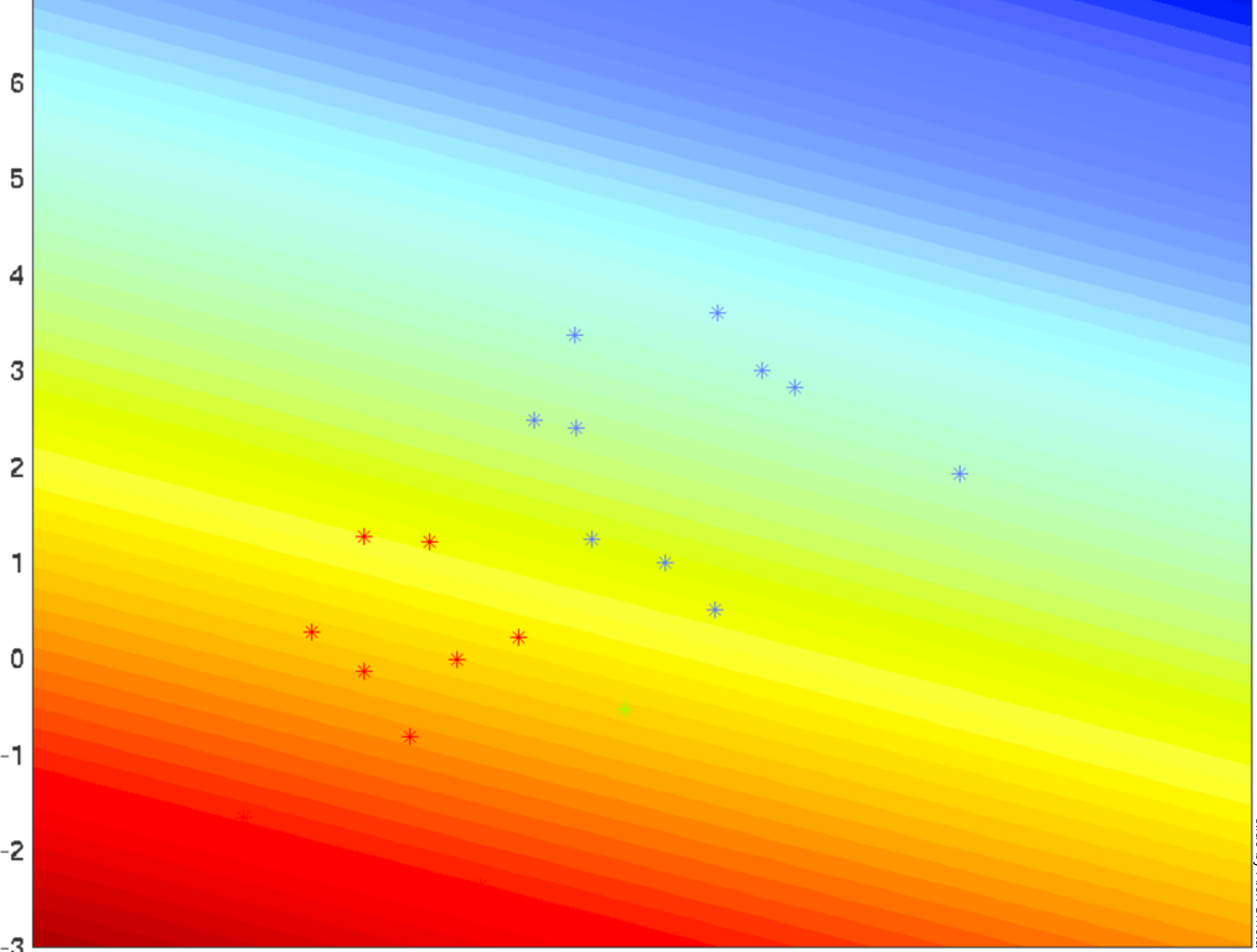

6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola

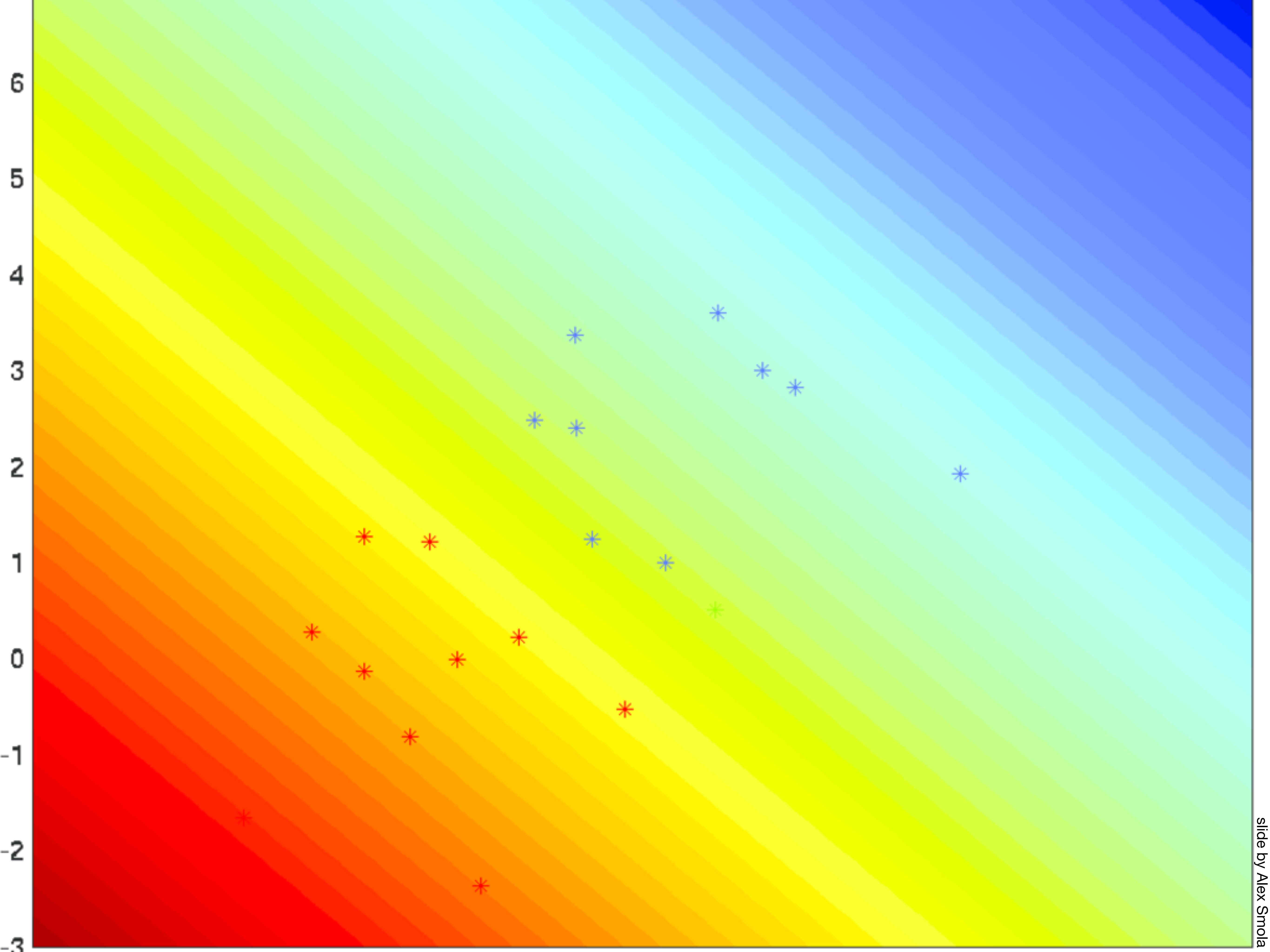

6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola

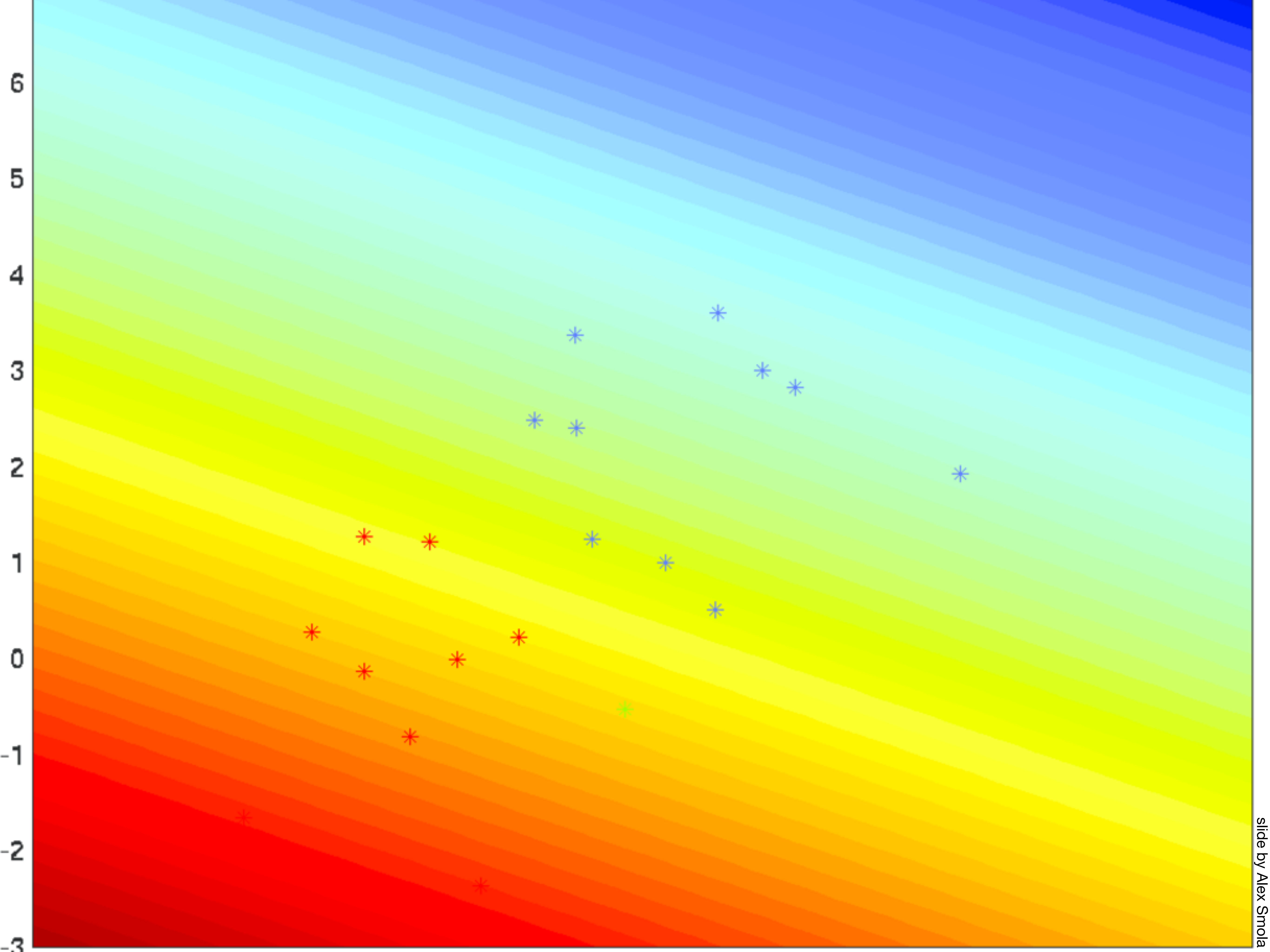

6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola

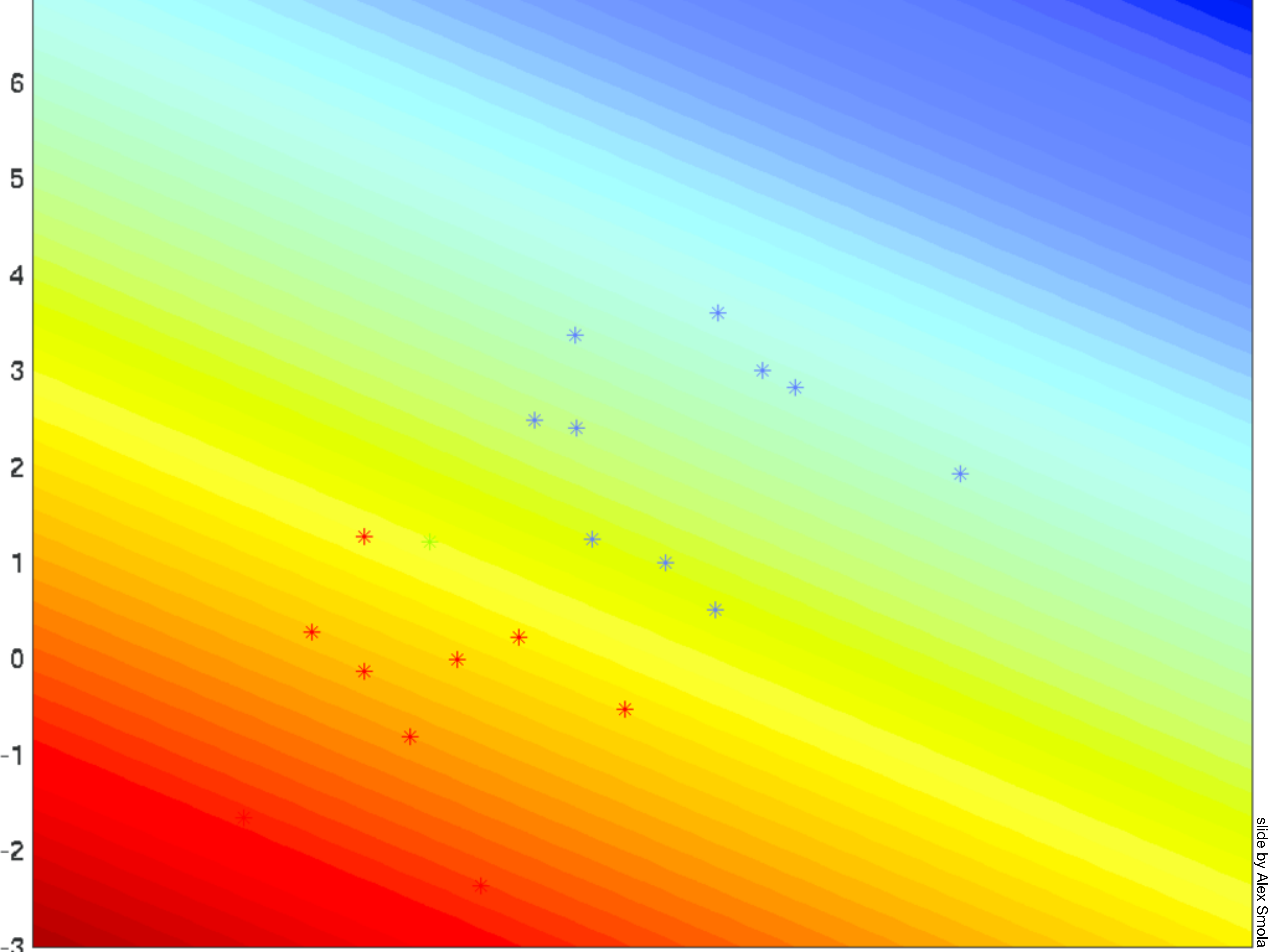

6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola


6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola


6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola

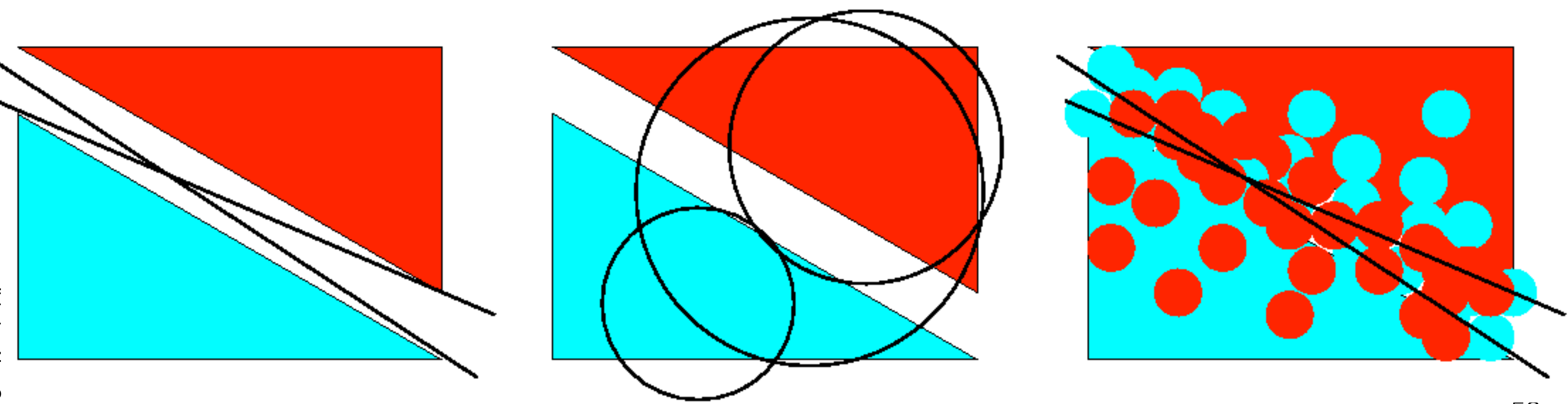

6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola


6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola


6
5
4
3
2
1
0
-1
-2
-3
slide by Alex Smola

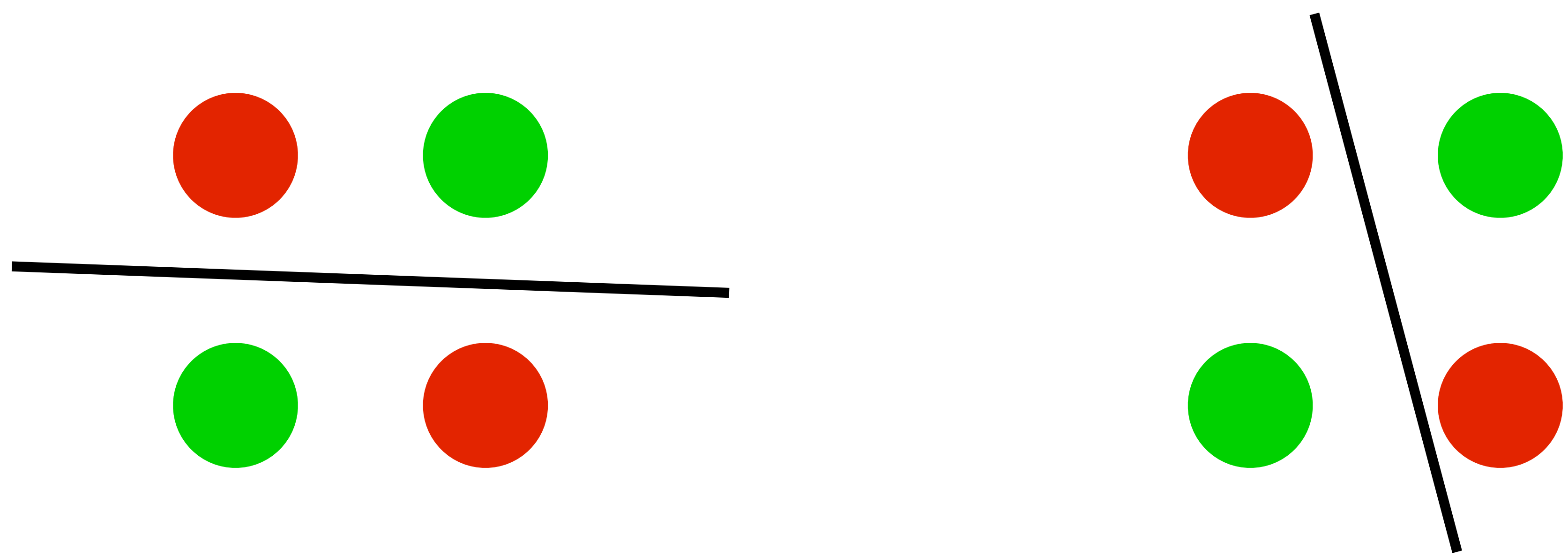
Concepts & version space

- 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)



slide by Alex Smola

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).

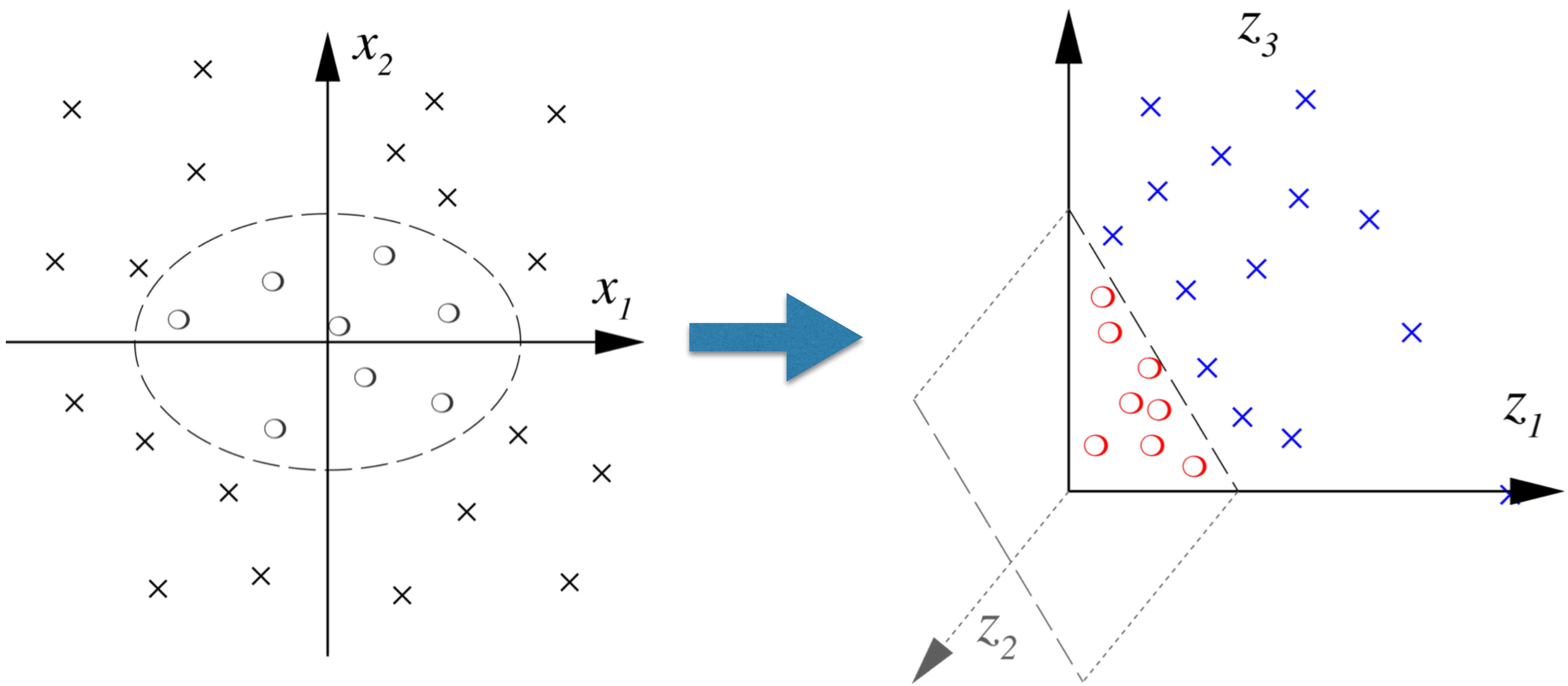
slide by Alex Smola

Nonlinear Features

- Regression
  We got nonlinear functions by preprocessing
- Perceptron
  - Map data into feature space $x \to \phi(x)$
  - 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 $\phi(x_i)$

slide by Alex Smola

Quadratic Features



- Separating surfaces are
Circles, hyperbolae, parabolae

slide by Alex Smola

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

slide by Alex Smola

Feature Engineering for Spam Filtering

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

```
Delivered-To: alex.smola@gmail.com
Received: by 10.216.47.73 with SMTP id s51cs361171web;
        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>
Received: from mail-ey0-f175.google.com (mail-ey0-f175.google.com [209.85.215.175])
        by mx.google.com with ESMTPS id n4si29264232eef.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) 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)
smtp.mail=alex+caf_=alex.smola=gmail.com@smola.org; dkim=pass (test mode)
header.i=@googlemail.com
Received: by eaal1 with SMTP id l1so15092746eaa.6
        for <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
X-Forwarded-For: alex@smola.org alex.smola@gmail.com
Delivered-To: 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: <althoff.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 althoff.tim@googlemail.com designates
209.85.220.179 as permitted sender) client-ip=209.85.220.179;
Received: by vcbf13 with SMTP id f13so11295098vcb.10
        for <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=WCbdZ5sXac25dpH02XcRyDOdts993hKwsAVXpGrFh0w=;
        b=WK2B2+ExWnf/gvTkw6uUvKuP4XeoKnlJq3USYTm0RARK8dSFjyOQsIHeAP9Yssxp6O
         7ngGoTzYqd+ZsyJfvQcLAWp1PCJhG8AMcnqWkx0NMeoFvIp2HQooZwxSOCx5ZRgY+7qX
         uIbbdna4lUDXj6UFe16SpLDCkptd8OZ3gr7+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: althoff.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: 6bwi6D17HjZIkxOEol38NZzyeHs
Message-ID: <CAFJJHDGPBW+SdZg0MdAABiAKydDk9tpeMoDijYGjoGO-WC7osg@mail.gmail.com>
Subject: CS 281B. Advanced Topics in Learning and Decision Making
From: Tim Althoff <althoff@eecs.berkeley.edu>
To: alex@smola.org
Content-Type: multipart/alternative; boundary=f46d043c7af4b07e8d04b5a7113a

--f46d043c7af4b07e8d04b5a7113a
Content-Type: text/plain; charset=ISO-8859-1
```

slide by Alex Smola

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
- Preprocessing
  - Zero mean, unit variance to fix scale issue (e.g. weight vs. income)
  - Probability integral transform (inverse CDF) as alternative

slide by Alex Smola

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

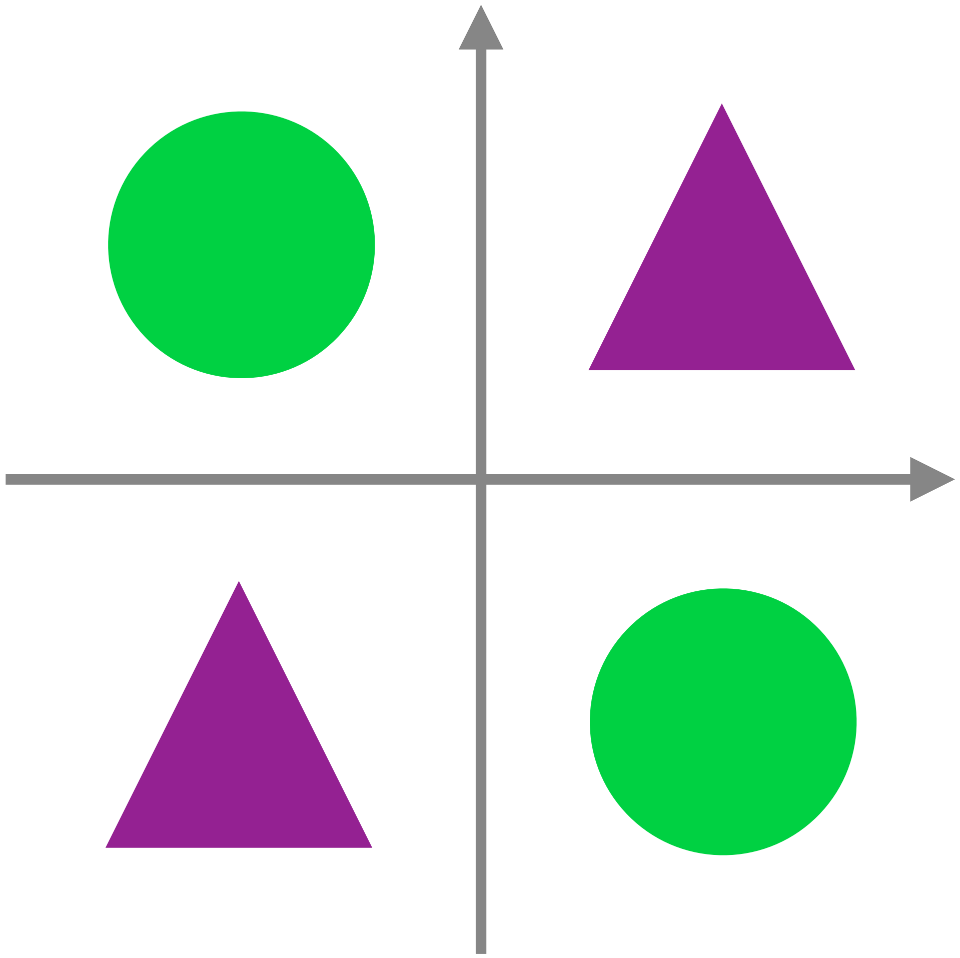
$$\begin{aligned} w' &= w + y_i \Phi(x_i) \\ b' &= b + y_i \end{aligned}$$

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

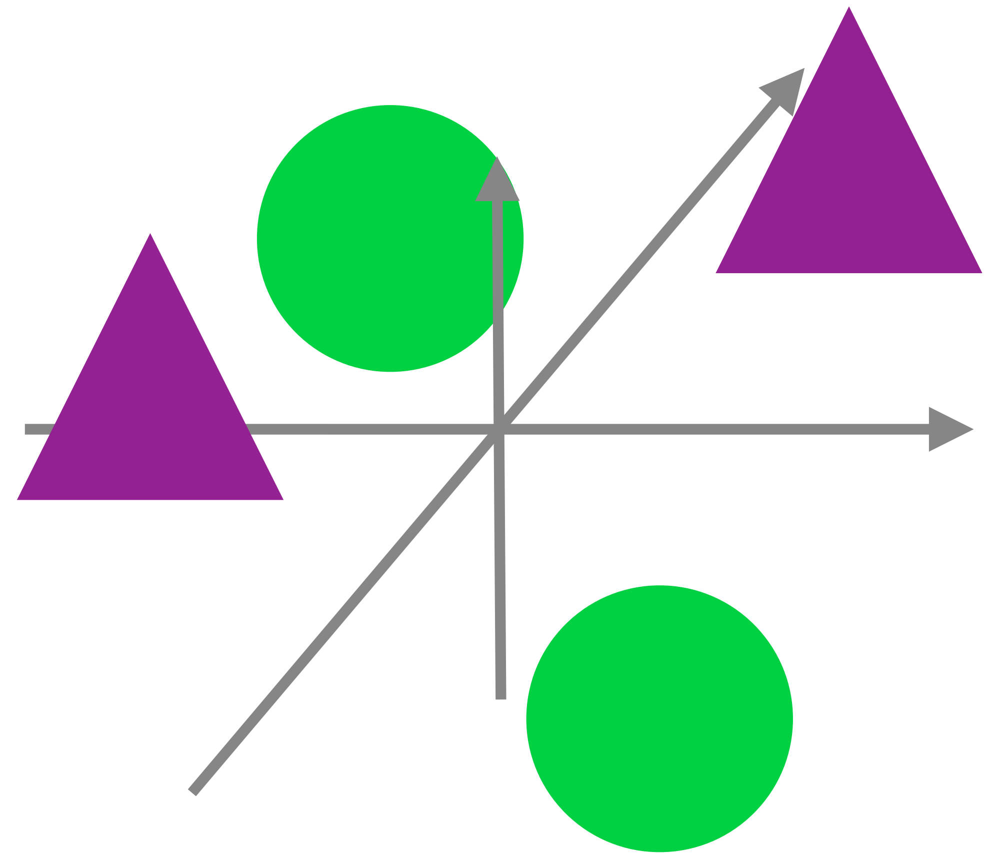
- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b$

slide by Alex Smola

Solving XOR



(x_1, x_2)



$(x_1, x_2, x_1 x_2)$

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

slide by Alex Smola

Extensions of Perceptron

- **Voted Perceptron**
  - generalizes better than (standard) perceptron
  - memory intensive (keeps around every weight vector seen during training, so each one can vote)
- **Averaged Perceptron**
  - empirically similar performance to voted perceptron
  - can be implemented in a memory efficient way (running averages are efficient)

slide by Matt Gormley & Henry Chai

Extensions of Perceptron

- **Kernel Perceptron**
  - Choose a kernel K(x’, x)
  - Apply the **kernel trick** to Perceptron
  - Resulting algorithm is still **very simple**
- **Structured Perceptron**
  - Basic idea can also be applied when **y** ranges over an exponentially large set
  - Mistake bound **does not** depend on the size of that set

slide by Matt Gormley & Henry Chai

Summary: Perceptron

- Perceptron is a **linear classifier**
- **Simple learning algorithm:** when a mistake is made, add / subtract the features
- Perceptron will converge if the data are **linearly separable**, it will not converge if the data are **linearly inseparable**
- For linearly separable and inseparable data, we can **bound the number of mistakes** (geometric argument)
- **Extensions** support nonlinear separators and structured prediction

slide by Matt Gormley & Henry Chai

Next Lecture:

Multi-layer Perceptron