Neural Networks

Thomas Stibor

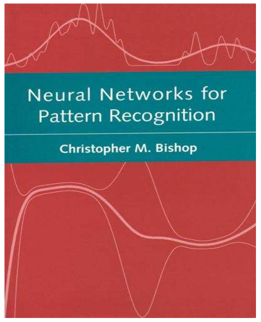
stibor@sec.informatik.tu-darmstadt.de

Department of Computer Science
Darmstadt University of Technology
Germany

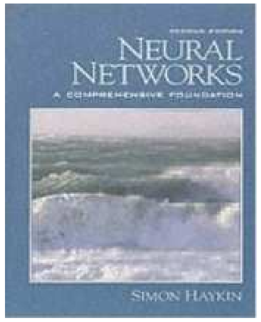
Topics Overview

- Single-Layer Networks
- Multi-Layer Networks
- Error Functions
- Weights Optimization Algorithms and Regularization
- Hopfield Network
- Boltzmann Machine
- Winner Take All Neural Network (Unsupervised Learning)
- Evolving Neural Networks
- Comparison to Support Vector Machine

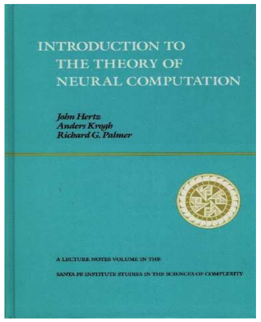
Literature - Books



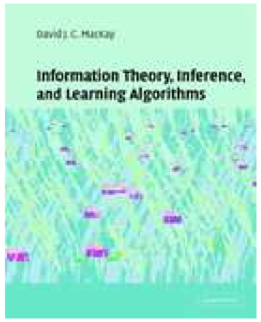
Neural Networks
for Pattern Recognition
Christopher M. Bishop
Oxford University Press, 1995



Neural Networks:
A Comprehensive Foundation (2.ed)
Simon Haykin
Prentice Hall Publishers, 1998



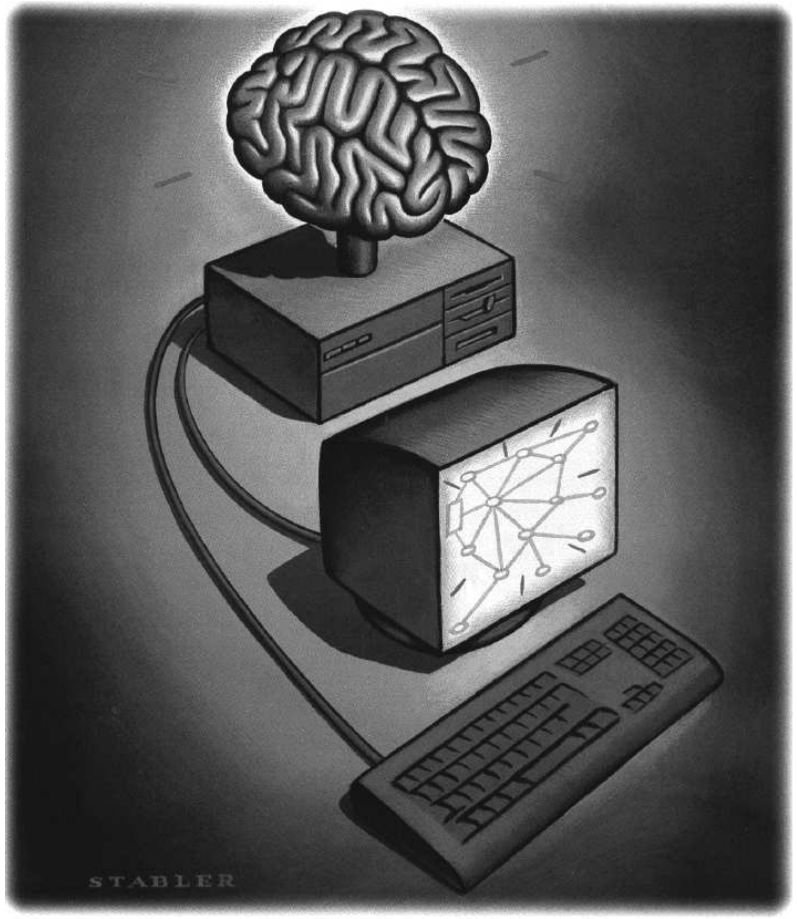
Introduction to the Theory
of Neural Computation
John Hertz, Andreas Krogh
and Richard Palmer
Addison Wesley, 1991



Information Theory, Inference
and Learning Algorithms
David MacKay,
Cambridge University Press, 2003

Some figures are taken from Bishop's new book (Pattern Recognition and Machine Learning).

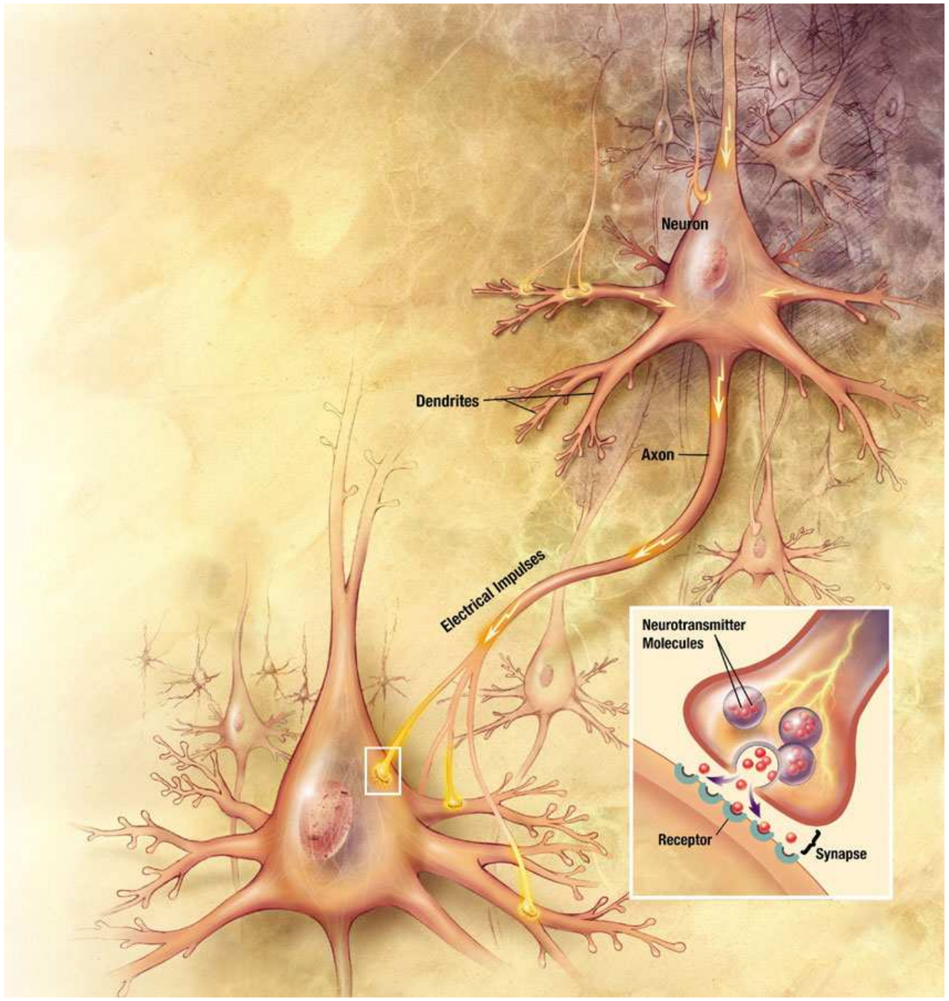
Neural Computation



Some appealing features of the brain that would be desirable in a computational model:

- Robust and fault tolerant
- Easily adjust to a new environment by “learning”
- Deal with information that is fuzzy, probabilistic, noisy, or inconsistent
- Highly parallel

Components of a Neuron



Brain is composed of about 10^{11} neurons. Each neuron has on average 7000 synaptic connections to other neurons. Neurons processing and transmitting information (electrical signals).

Inspiration from Neuroscience

In early days term “neural network” was motivated towards modelling networks of real neurons in the brain.

“The perspective of statistical pattern recognition, however, offers a much more direct and principled route to many of the same concepts.” Christopher M. Bishop [Neural Networks for Pattern Recognition]

In this course neural networks are presented from this perspective (computational geometry, statistics, optimization).

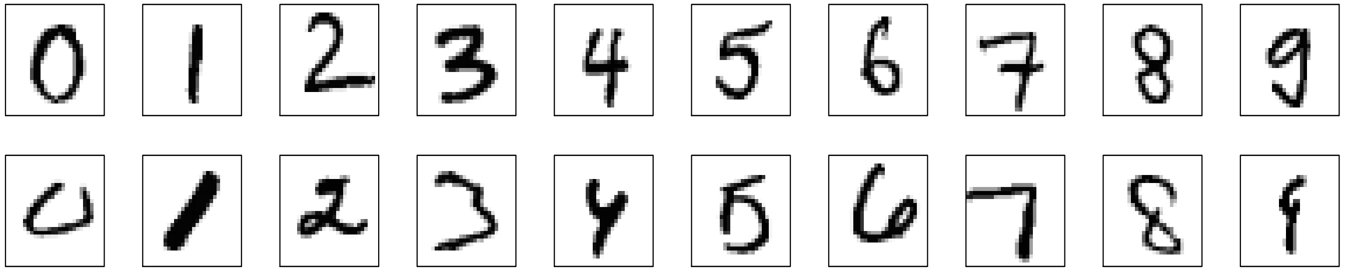
History of Neural Networks

	Era of Kernel Methods (SVM, Kernel-PCA, Kernel-Fisher Discriminants, etc.) **Neural Networks are however still and frequently used**
1995 ●	Support-Vector Networks (Cortes and Vapnik)
1992 ●	A Training Algorithm for Optimal Margin Classifiers (Boser, Guyon and Vapnik), first paper on SVM
	Era of Neural Networks
1986/ 1985 ●	Backpropagation (Rumelhart, Hinton, Williams, Le Cun (actually first proposed by Werbos, 1974))
1982 ●	Hopfield Network, (Hopfield), Recurrent Networks, Energy Function
	Decline of neural network research
1969 ●	Book: Perceptrons (Minsky and Papert)
1962/ 1960 ●	Adaline (Widrow and Hoff), Perceptron (Rosenblatt)
1943 ●	Model of McCulloch and Pitts

Note, this historical overview is far from being complete (see books for detailed historical overview)

Motivation (Application of Neural Networks)

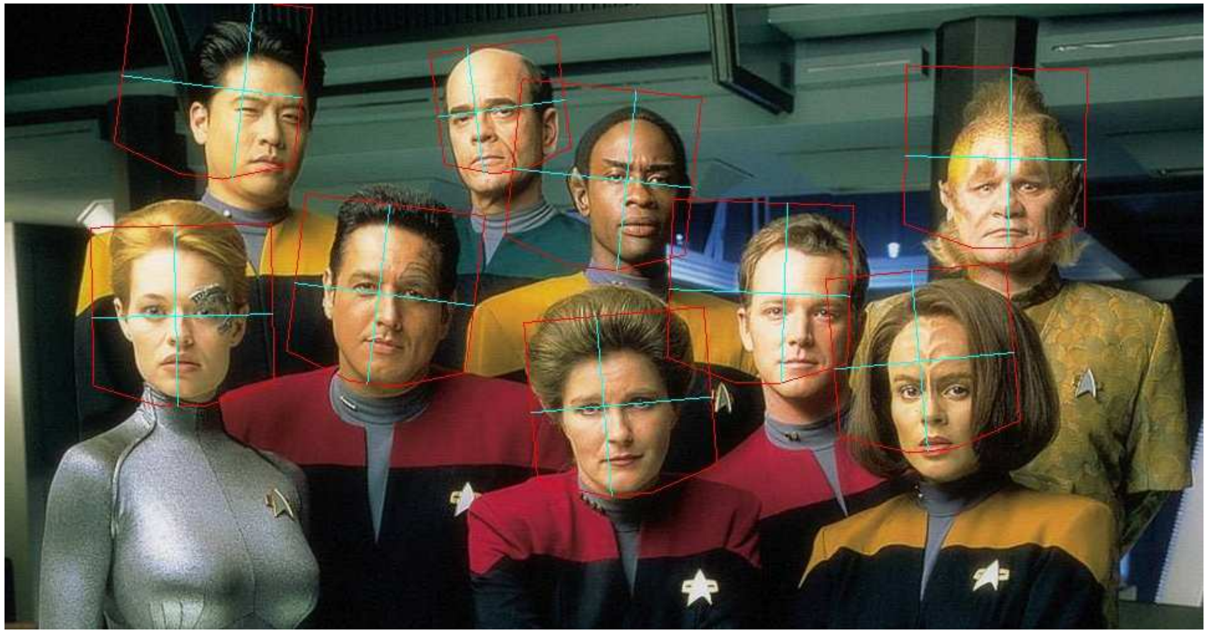
- Handwritten Digit Recognition



- Digits are size-normalized and centered in a 28×28 fixed-size image of gray color values $(0 - 255)$
- Given a vector $\mathbf{x} = [0, 0, 0, \dots 67, 114, 72, \dots 0, 0, 0] \in \{0, \dots, 255\}^{784}$ which represents a new (unseen) digit, to which digit class belongs $\mathbf{x}$?

Motivation (Application of Neural Networks)

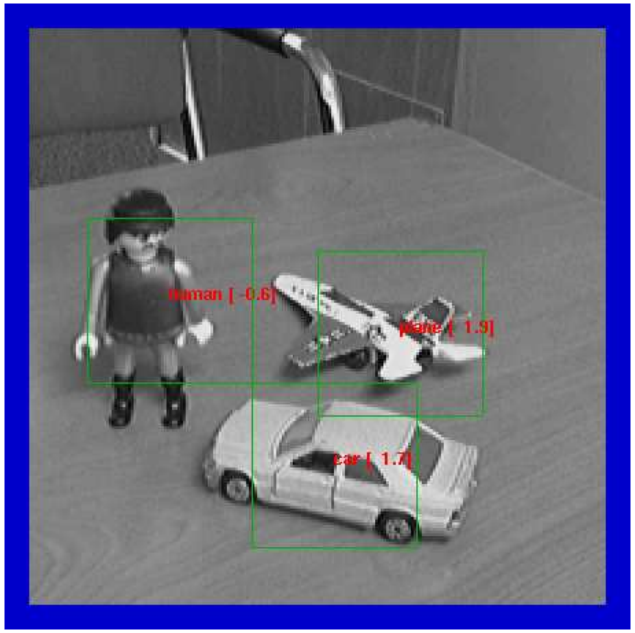
- Face Detection

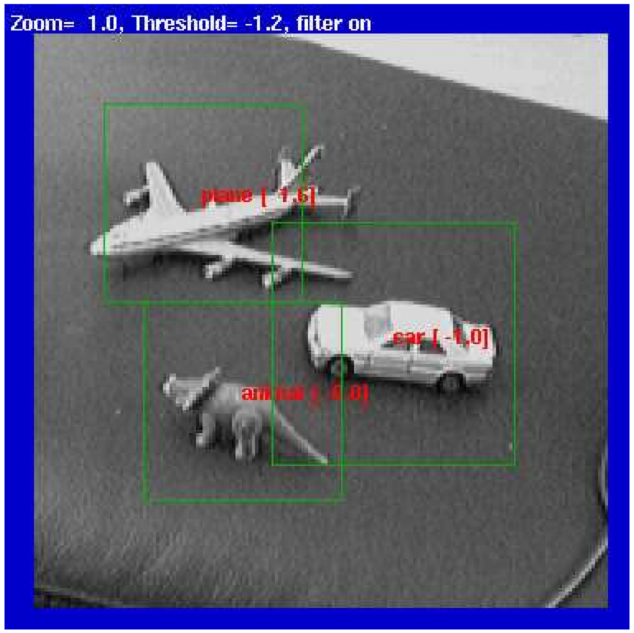


see website of Yann LeCun
(http://www.cs.nyu.edu/~yann/research/cface/)

Motivation (Application of Neural Networks)

- Object Detection and Recognition

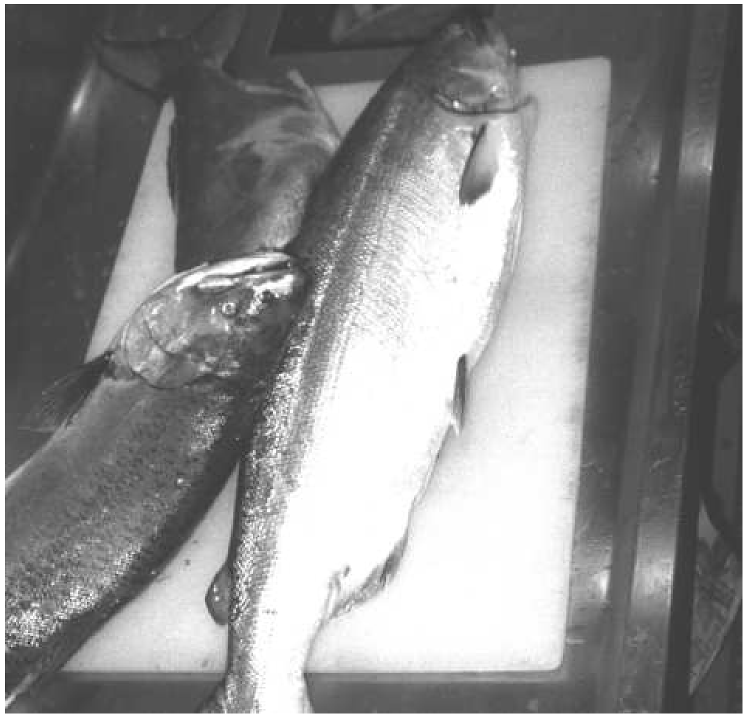


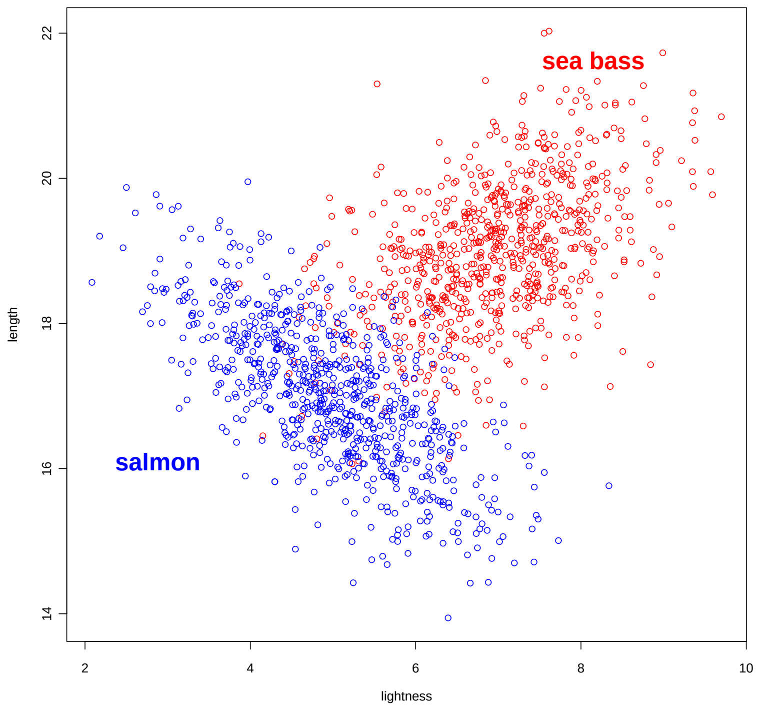


see website of Yann LeCun
(http://www.cs.nyu.edu/∼yann/research/norb/)

Motivation (cont.)

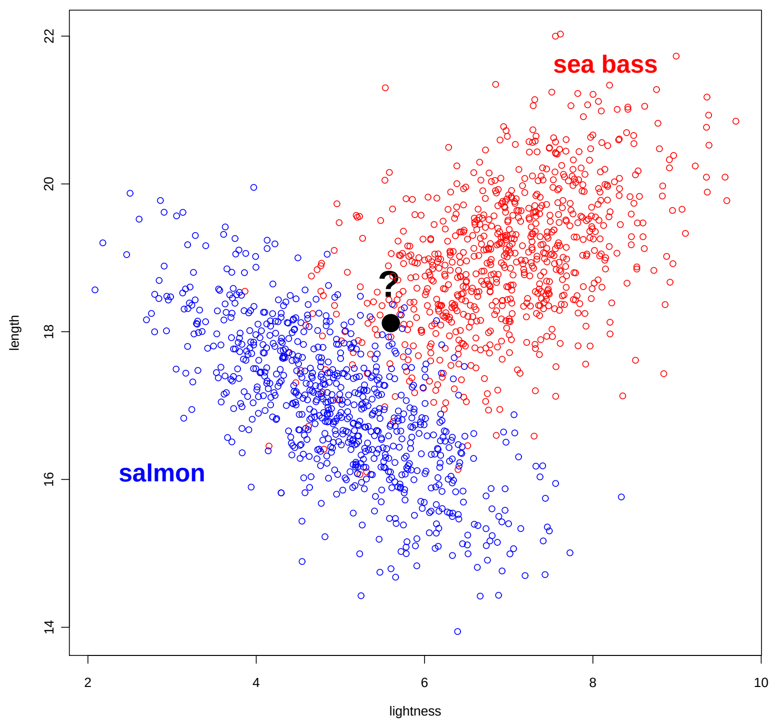
Suppose that a fishpacking factory wants to automate the process of sorting incoming fish (salmon and sea bass).





After some preprocessing, each fish is characterized by feature vector $\mathbf{x} = [x_1, x_2]$ (pattern), where the first component is the lightness and the second component the length.

Pattern belongs to Class?



Given labeled training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathbb{R}^d \times Y$ coming from some unknown probability distribution $p(\mathbf{x}, y)$. In this example, $Y = \{\text{salmon}, \text{sea bass}\}$ and $d = 2$. Unseen (unlabeled) pattern belongs to class salmon or sea bass?

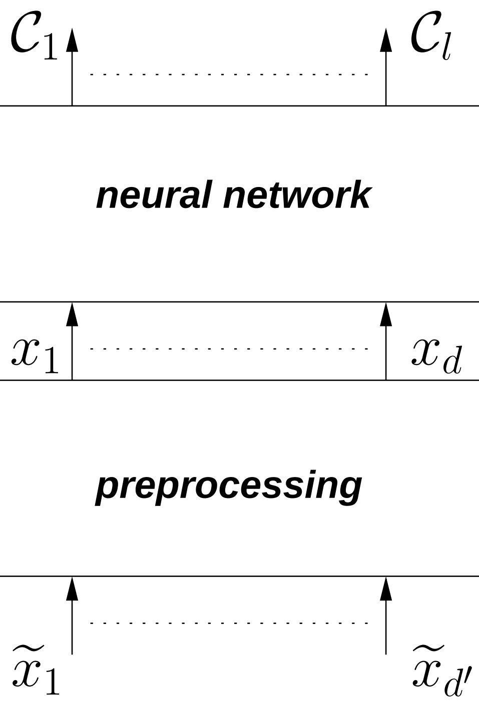
Classes of Learning Algorithms

Pattern classification algorithms can be divided roughly in *two classes*.

- **Supervised learning** (learning with a teacher)
  - Training data consists of class labels
  - Output is compared to target output and learning parameters (e.g. weights in NN) are corrected according to the magnitude of the error
- **Unsupervised learning**
  - No class labels are available
  - Cluster patterns according to some similarity measure (e.g. Euclidean distance)

Neural Network Classification Model

We are looking for a classification method which *learns* from training data (available examples) and outputs a class membership for *unseen* data (testing data).



For performing classification we need:

- Similarity measure
- Discrimination function

Dot Product as a Similarity Measure

Dot product of two vectors $\mathbf{a} = (a_1, \ldots, a_n)$, $\mathbf{b} = (b_1, \ldots, b_n)$:

$$(\mathbf{a}^T \cdot \mathbf{b}) = a_1 b_1 + a_2 b_2 + \ldots a_n b_n = \sum_{i=1}^{n} a_i b_i; \quad \text{(short form } \mathbf{a}^T\mathbf{b}\text{)}$$

Dot product allow us to compute: lengths, angles, distances.
Length (norm):

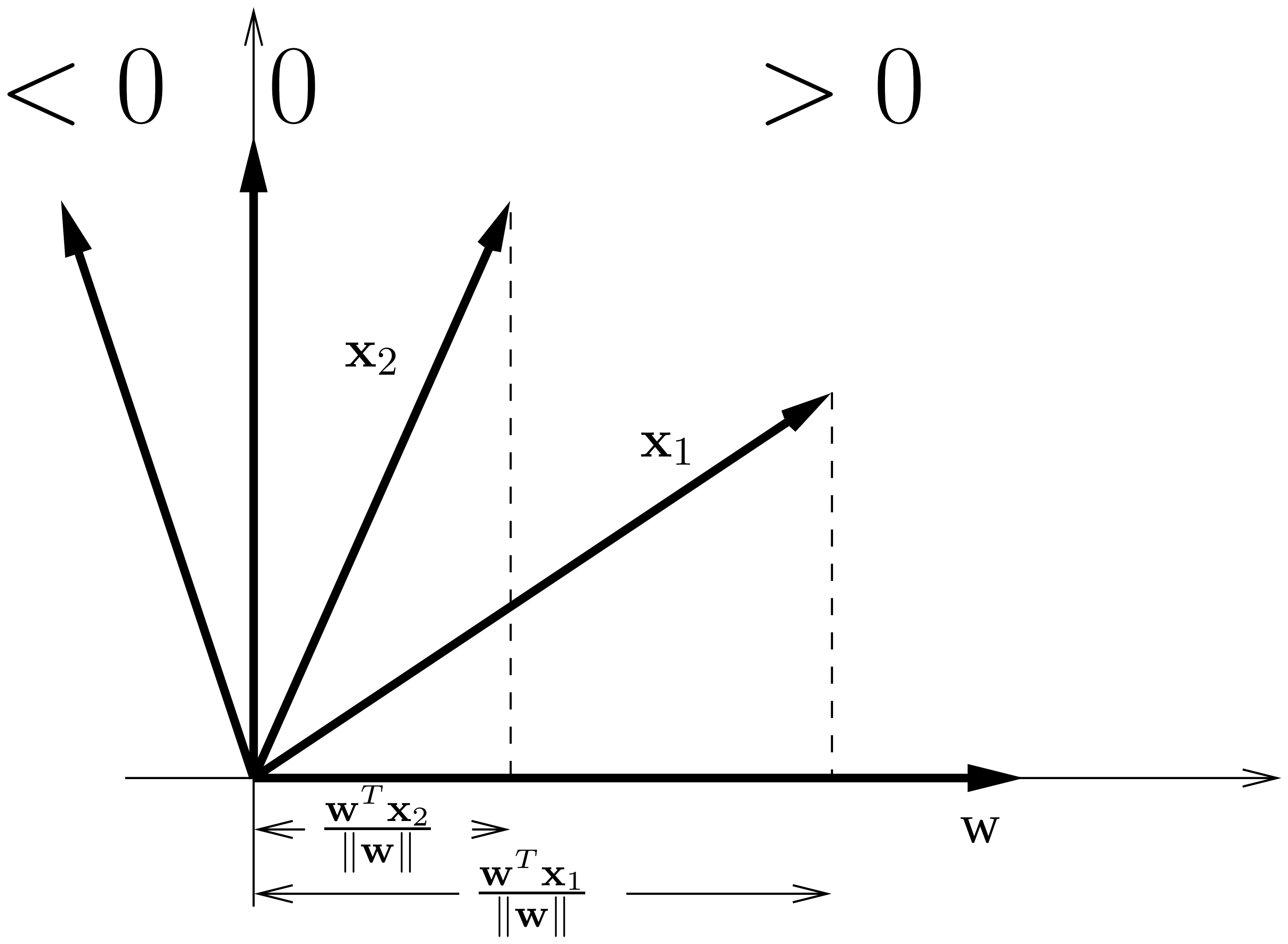
$$\|\mathbf{a}\|^2 = a_1 a_1 + a_2 a_2 + \ldots + a_n a_n = (\mathbf{a}^T \mathbf{a})$$

Example: $\mathbf{a} = (1,1,1) \Rightarrow \|\mathbf{a}\| = \sqrt{1^2 + 1^2 + 1^2} = \sqrt{3}$

Angle:

$$\cos\phi = \frac{(\mathbf{a}^T\mathbf{b})}{\|\mathbf{a}\|\,\|\mathbf{b}\|} = \frac{a_1 b_1 + a_2 b_2 + \ldots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}\sqrt{b_1^2 + b_2^2 + \ldots + b_n^2}}$$

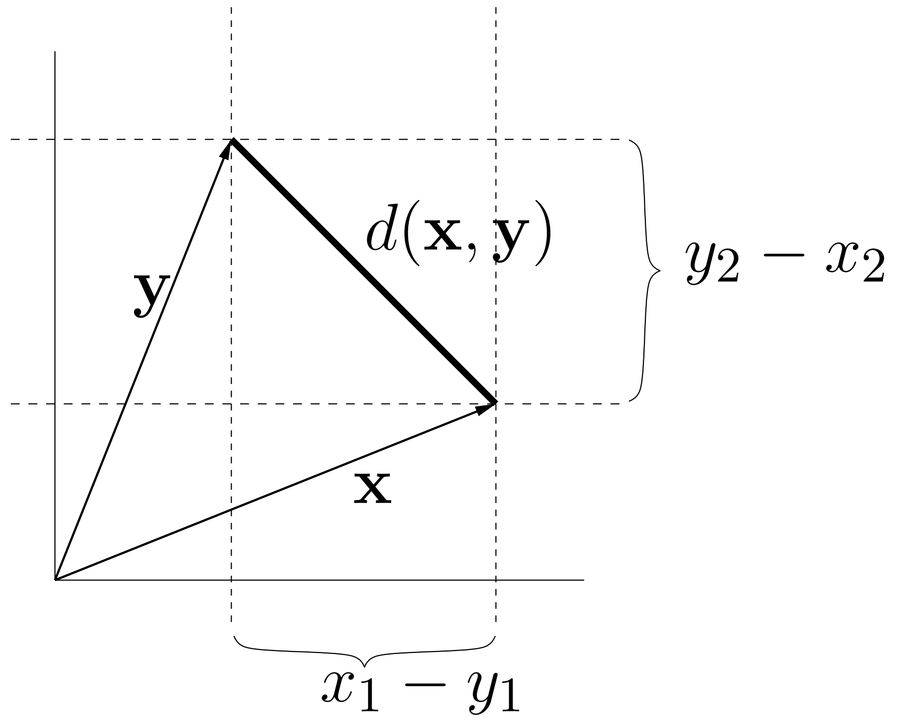
Dot Product as Similarity Measure (Projection)



< 0
0
> 0
$\mathbf{x}_2$
$\mathbf{x}_1$
$\mathbf{w}$
$\frac{\mathbf{w}^T\mathbf{x}_2}{\|\mathbf{w}\|}$
$\frac{\mathbf{w}^T\mathbf{x}_1}{\|\mathbf{w}\|}$

Geometric interpretation: Length of the projection of $\mathbf{x}$ onto the unit vector $\mathbf{w}/\|\mathbf{w}\|$.

Dot Product (Distances)



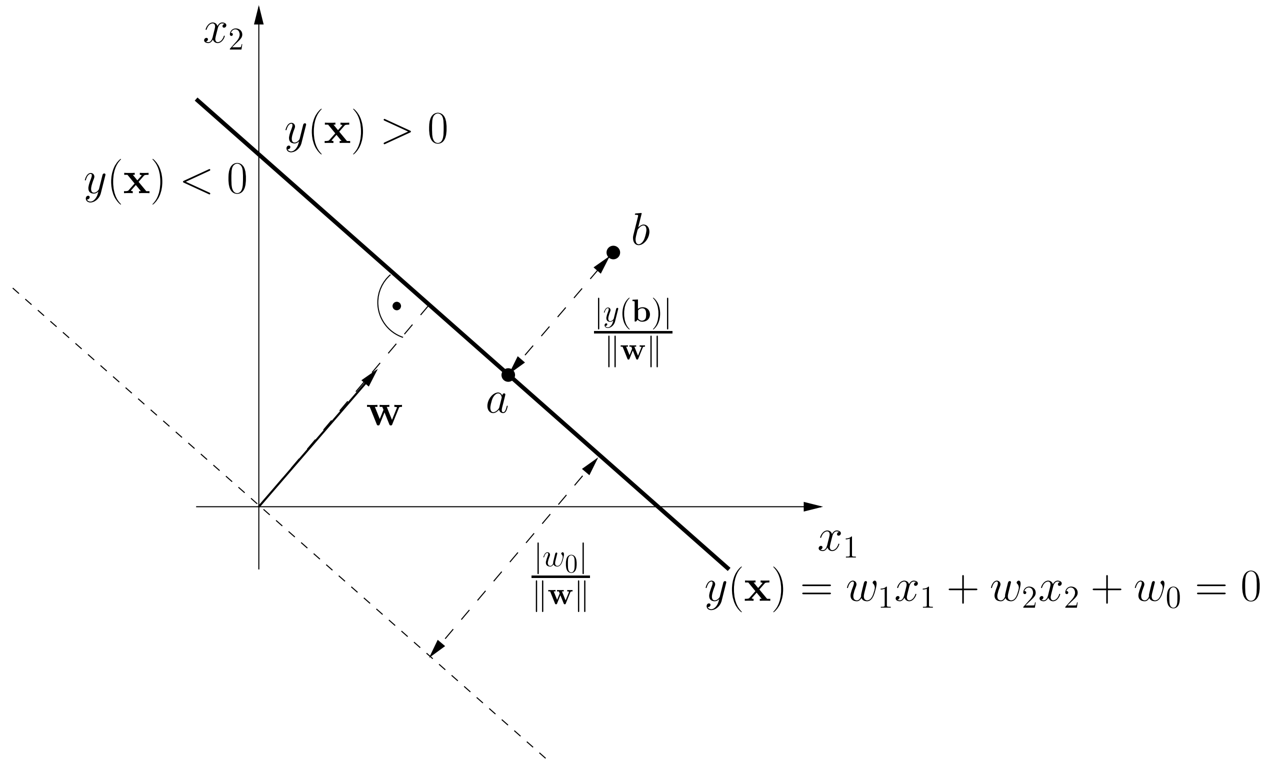
$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$$

$$(d(\mathbf{x}, \mathbf{y}))^2 = \|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\left(\mathbf{x}^T \mathbf{y}\right)$$

Discrimination Function

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

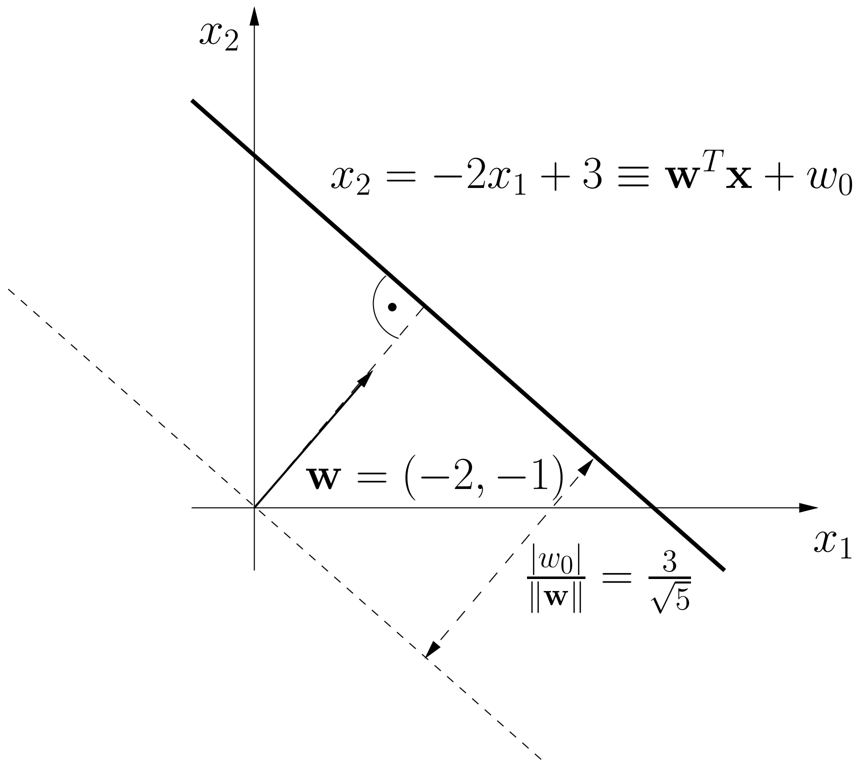
where $\mathbf{w}$ is d-dim. *weight vector* and w_0 the bias (threshold).



x_2
$y(\mathbf{x}) > 0$
$y(\mathbf{x}) < 0$
b
$\frac{|y(\mathbf{b})|}{\|\mathbf{w}\|}$
$\mathbf{w}$
a
x_1
$\frac{|w_0|}{\|\mathbf{w}\|}$
$y(\mathbf{x}) = w_1x_1 + w_2x_2 + w_0 = 0$

Decision boundary $y(\mathbf{x}) = 0$ corresponds to a $(d-1)$-dim. hyperplane in d-dim. $\mathbf{x}$-space. For $d = 2$ it is a straight line.

Discrimination Function Example


x_2
x_1
$x_2 = -2x_1 + 3 \equiv \mathbf{w}^T\mathbf{x} + w_0$
$\mathbf{w} = (-2, -1)$
$\frac{|w_0|}{\|\mathbf{w}\|} = \frac{3}{\sqrt{5}}$

$$x_2 = -2x_1 + 3 \Leftrightarrow -2x_1 - 1x_2 + 3 \equiv (w_1, w_2)^T\mathbf{x} + w_0 = 0.$$

Vector $\mathbf{w}$ defines the orientation of the decision plane, bias w_0 the position in terms of its perpendicular distance from the origin.

Distances and Optimization

We introduce the method of Lagrange multipliers and gradients to get an intuitive understanding for the further lectures (gradient descent in neural networks and Lagrange multipliers in SVMs).

Problem: find point on plane ($d = 2$, line) that is closest to the origin, in other words, derive $\frac{|w_0|}{\|\mathbf{w}\|}$.

Distance from origin to a point $\mathbf{a} = (a_1, a_2)$ on the line is $d(\mathbf{a}, \mathbf{0}) = \sqrt{a_1^2 + a_2^2}$ with constraint $w_1 a_1 + w_2 a_2 + w_0 = 0$, that is:

$$\begin{array}{rl} \text{minimize} & \sqrt{a_1^2 + a_2^2} \\ \text{subject to} & w_1 a_1 + w_2 a_2 + w_0 = 0 \end{array}$$

Distances and Optimization (cont.)

Lagrangian $L(a_1, a_2, \alpha) = \sqrt{a_1^2 + a_2^2} - \alpha(w_1 a_1 + w_2 a_2 + w_0)$

$$\begin{aligned}
\frac{\partial L(a_1, a_2, \lambda)}{\partial a_1} &= \frac{a_1}{\sqrt{a_1^2 + a_2^2}} - \alpha w_1 = 0 \\
\frac{\partial L(a_1, a_2, \lambda)}{\partial a_2} &= \frac{a_2}{\sqrt{a_1^2 + a_2^2}} - \alpha w_2 = 0 \\
\frac{\partial L(a_1, a_2, \lambda)}{\partial \alpha} &= -w_1 a_1 - w_2 a_2 - w_0 = 0
\end{aligned}$$

Solving this eq.-system gives:

$$a_1 = \frac{-w_1 w_0}{w_1^2 + w_2^2}, \; a_2 = \frac{-w_2 w_0}{w_1^2 + w_2^2}$$

Distances and Optimization (cont.)

Substituting back in $d(\mathbf{a}, \mathbf{0})$ gives:

$$\sqrt{\frac{(-w_1 w_0)^2}{(w_1^2 + w_2^2)^2} + \frac{(-w_2 w_0)^2}{(w_1^2 + w_2^2)^2}} = \sqrt{\frac{w_0^2(w_1^2 + w_2^2)}{(w_1^2 + w_2^2)^2}} = \frac{|w_0|}{\|\mathbf{w}\|}$$

Problem: find point $\mathbf{a}$ on the line that is closest to a point $\mathbf{b}$, $(w_1 b_1 + w_2 b_2 + w_0 \neq 0)$, in other words, show

$$d(\mathbf{a}, \mathbf{b}) = \frac{w_1 b_1 + w_2 b_2 + w_0}{\|\mathbf{w}\|} = \frac{|y(\mathbf{b})|}{\|\mathbf{w}\|}$$

$$L(a_1, a_2, \alpha) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} - \alpha(w_1 a_1 + w_2 a_2 + w_0)$$

Distances and Optimization (cont.)

$$\begin{aligned}
\frac{\partial L(a_1, a_2, \lambda)}{\partial a_1} &= \frac{a_1 - b_1}{\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}} - \alpha w_1 = 0 \\
\frac{\partial L(a_1, a_2, \lambda)}{\partial a_2} &= \frac{a_2 - b_2}{\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}} - \alpha w_2 = 0 \\
\frac{\partial L(a_1, a_2, \lambda)}{\partial \alpha} &= -w_1 a_1 - w_2 a_2 - w_0 = 0
\end{aligned}$$

Solving this eq.-system gives:

$$\begin{aligned}
a_1 &= \frac{b_1 w_2^2 - w_1 w_2 b_2 - w_1 w_0}{w_1^2 + w_2^2} \\
a_2 &= \frac{b_2 w_1^2 - w_1 w_2 b_1 - w_2 w_0}{w_1^2 + w_2^2}
\end{aligned}$$

Distances and Optimization (cont.)

Substituting back in $d(\mathbf{a}, \mathbf{b})$ gives:

$$\left[\left(\frac{-b_1 w_1^2 - w_1 w_2 b_2 - w_1 w_0}{w_1^2 + w_2^2}\right)^2 + \left(\frac{-b_2 w_2^2 - w_1 w_2 b_2 - w_2 w_0}{w_1^2 + w_2^2}\right)^2\right]^{1/2} =$$

$$\left[\frac{w_1^2(-b_1 w_1 - w_2 b_2 - w_0)^2 + w_2^2(-b_2 w_2 - w_1 b_1 - w_0)^2}{(w_1^2 + w_2^2)^2}\right]^{1/2} =$$

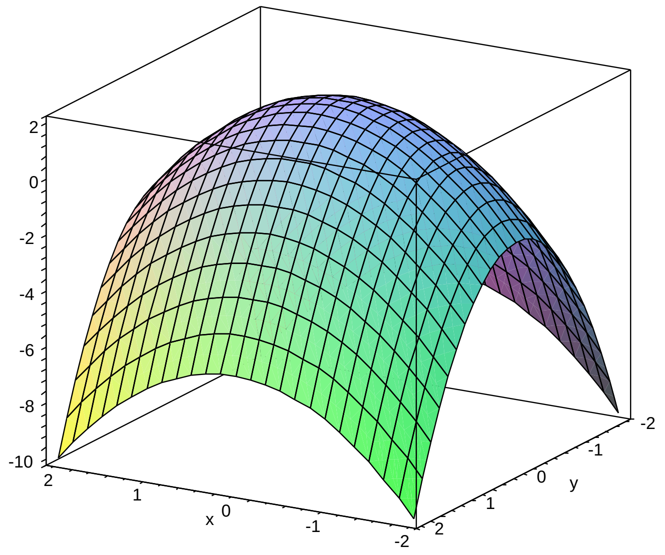
$$\sqrt{\frac{(w_1^2 + w_2^2)(w_1 b_1 + w_2 b_2 + w_0)^2}{(w_1^2 + w_2^2)^2}} = \frac{|w_1 b_1 + w_2 b_2 + w_0|}{\|\mathbf{w}\|} = \frac{|y(\mathbf{b})|}{\|\mathbf{w}\|}$$

Lagrange multipliers

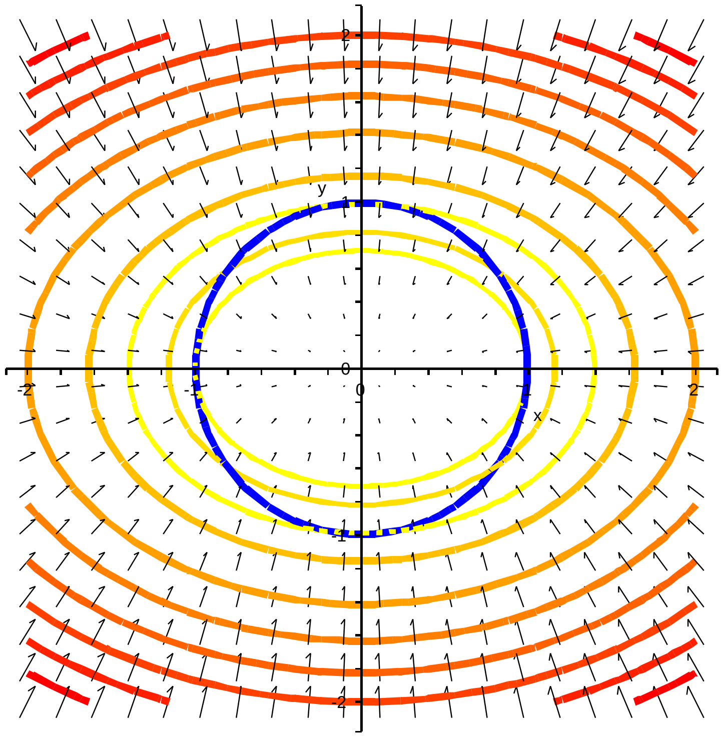
Given the following optimization problem:

$$\begin{aligned} \text{maximize} \qquad & f(x,y) = 2 - x^2 - 2y^2 \\ \text{subject to} \qquad & g(x,y) = x^2 + y^2 - 1 = 0. \end{aligned}$$

With *Lagrange multipliers* we can find the extrema of a function of several variables subject to one or more constraints.



Lagrange multipliers (cont.)



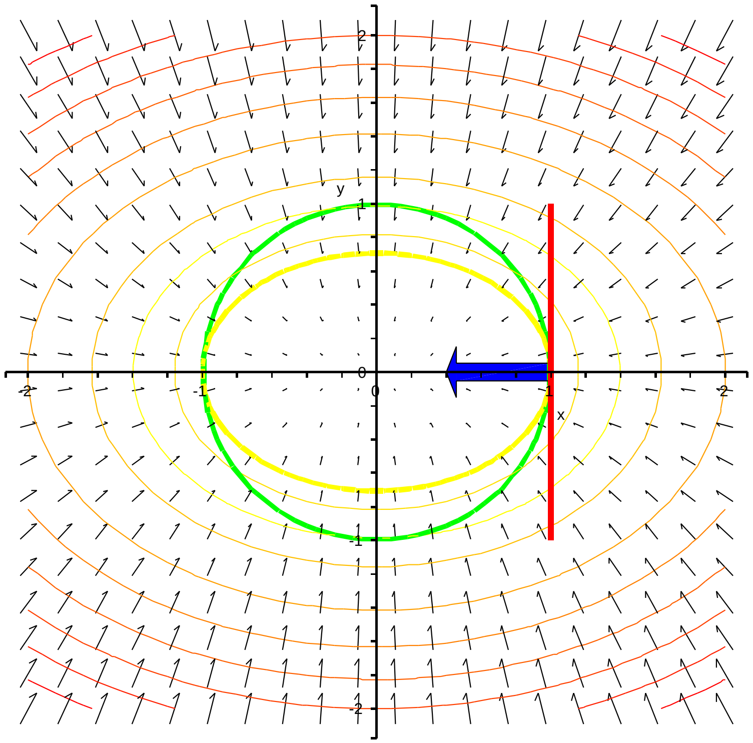
The gradient of f,

$$\begin{aligned}\nabla f &= \text{grad } f(\mathbf{x}) \\ &= \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)\end{aligned}$$

is a vector field, where the vectors point in the directions of the greatest increase of f.

The direction of greatest increase is always perpendicular to the level curves. The circle (blue curve) is the feasible region satisfying the constraint $x^2 + y^2 - 1 = 0$

Lagrange multipliers (cont.)



At extreme points (x, y) the gradients of f and g are parallel vectors, that is

$$\nabla f(x, y) = \lambda \nabla g(x, y)$$

To find the p_i we have to solve

$$\nabla f(x, y) - \lambda \nabla g(x, y) = 0$$

Lagrange multipliers Ex. 1

Back to our optimization problem:

$$\begin{array}{ll} \text{maximize} & f(x,y) = 2 - x^2 - 2y^2 \\ \text{subject to} & g(x,y) = x^2 + y^2 - 1 = 0. \end{array}$$

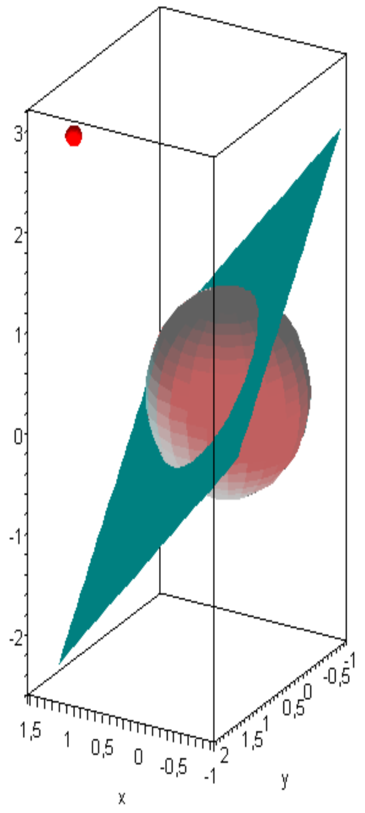
$$L(x,y,\lambda) = f(x,y) - \lambda g(x,y) = 2 - x^2 - 2y^2 - \lambda(x^2 + y^2 - 1)$$

$$\begin{aligned} \frac{\partial L(x,y,\lambda)}{\partial x} &= -2x - 2\lambda x = 0 \\ \frac{\partial L(x,y,\lambda)}{\partial y} &= -4y - 2\lambda y = 0 \\ \frac{\partial L(x,y,\lambda)}{\partial \lambda} &= -x^2 - y^2 + 1 = 0 \end{aligned}$$

Solving the equation system gives: $x = \pm 1$ and $y = 0$ $(\lambda = -1)$ and $x = 0$ and $y = \pm 1$ $(\lambda = -2)$.

Lagrange multipliers Ex. 2

Find the point $\mathbf{p}_t$ on the circle formed by the intersection of the unit sphere with the plane $x+y+z=\frac{1}{2}$ that is closest to the point $\mathbf{p}_g=(1,2,3)$, i.e. $\min \|\mathbf{p}_g-\mathbf{p}_t\|^2 \mathrel{\widehat{=}} \min f(x,y,z)$



$$\begin{aligned} f(x,y,z) &= (x-1)^2+(y-2)^2+(z-3)^2 \\ g_1(x,y,z) &= x^2+y^2+z^2-1 \\ g_2(x,y,z) &= x+y+z-\frac{1}{2} \end{aligned}$$

Lagrange multipliers Ex. 2 (cont.)

$$L(x,y,z,\boldsymbol{\lambda}) = (x-1)^2 + (y-2)^2 + (z-3)^2$$

$$+\lambda_1\left(x^2+y^2+z^2-1\right) + \lambda_2\left(x+y+z-\frac{1}{2}\right)$$

$$\begin{aligned}
\frac{\partial L(x,y,z,\boldsymbol{\lambda})}{\partial x} &= 2(x-1) + 2\lambda_1 x + \lambda_2 = 0 \\
\frac{\partial L(x,y,z,\boldsymbol{\lambda})}{\partial y} &= 2(y-2) + 2\lambda_1 y + \lambda_2 = 0 \\
\frac{\partial L(x,y,z,\boldsymbol{\lambda})}{\partial z} &= 2(z-3) + 2\lambda_1 z + \lambda_2 = 0 \\
\frac{\partial L(x,y,z,\boldsymbol{\lambda})}{\partial \lambda_1} &= x^2+y^2+z^2-1 = 0 \\
\frac{\partial L(x,y,z,\boldsymbol{\lambda})}{\partial \lambda_2} &= x+y+z-\frac{1}{2} = 0
\end{aligned}$$

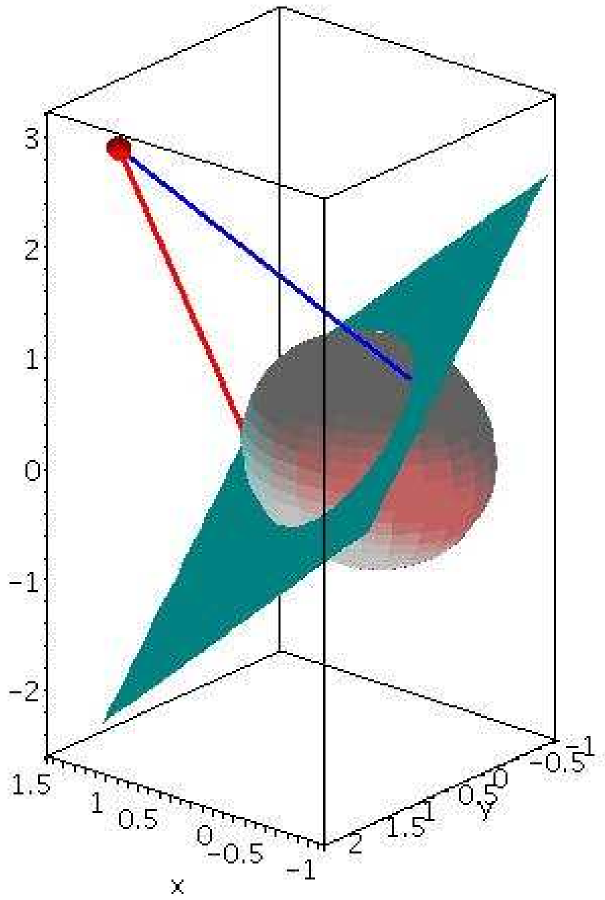
Lagrange multipliers Ex. 2 (cont.)

Solving this equation system gives:

$$\begin{aligned} x_1 &= \tfrac{1}{6} - \tfrac{1}{12}\sqrt{66}, & y_1 &= \tfrac{1}{6}, & z_1 &= \tfrac{1}{6} + \tfrac{1}{12}\sqrt{66} \\ x_1 &= -0.51, & y_1 &= 0.16, & z_1 &= 0.84 \end{aligned}$$

$$\begin{aligned} x_2 &= \tfrac{1}{6} + \tfrac{1}{12}\sqrt{66}, & y_2 &= \tfrac{1}{6}, & z_2 &= \tfrac{1}{6} - \tfrac{1}{12}\sqrt{66} \\ x_2 &= 0.84, & y_2 &= 0.16, & z_2 &= -0.51 \end{aligned}$$

Lagrange multipliers Ex. 2 (cont.)



Method of Steepest Descent

Let $E(\mathbf{w})$ be a continuously differentiable function of some unknown (weight) vector $\mathbf{w}$.

Find an optimal solution $\mathbf{w}^\star$ that satisfies the condition

$$E(\mathbf{w}^\star) \leq E(\mathbf{w}).$$

The necessary condition for optimality is

$$\nabla E(\mathbf{w}^\star) = \mathbf{0}.$$

Let us consider the following *iterative* descent:

Start with an initial guess $\mathbf{w}^{(0)}$ and generate sequence of weight vectors $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots$ such that

$$E(\mathbf{w}^{(i+1)}) \leq E(\mathbf{w}^{(i)}).$$

Steepest Descent Algorithm

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \nabla E(\mathbf{w}^{(i)})$$

where η is a positive constant called learning rate.

At each iteration step the algorithm applies the correction

$$\begin{aligned} \Delta \mathbf{w}^{(i)} &= \mathbf{w}^{(i+1)} - \mathbf{w}^{(i)} \\ &= -\eta \nabla E(\mathbf{w}^{(i)}) \end{aligned}$$

Steepest descent algorithm satisfies:

$$E(\mathbf{w}^{(i+1)}) \leq E(\mathbf{w}^{(i)}),$$

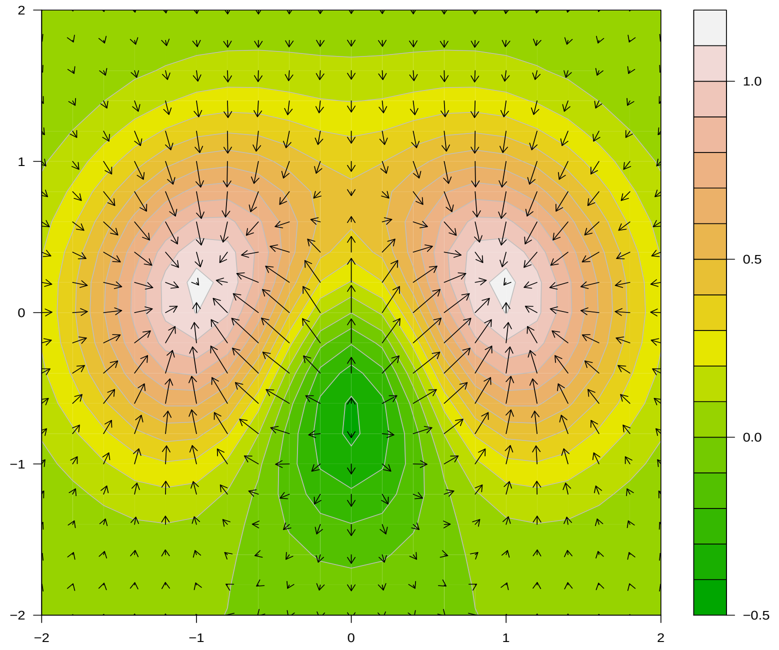
to see this, use first-order Taylor expansion around $\mathbf{w}^{(i)}$ to approximate $E(\mathbf{w}^{(i+1)})$ as $E(\mathbf{w}^{(i)}) + (\nabla E(\mathbf{w}^{(i)}))^T \Delta \mathbf{w}^{(i)}$.

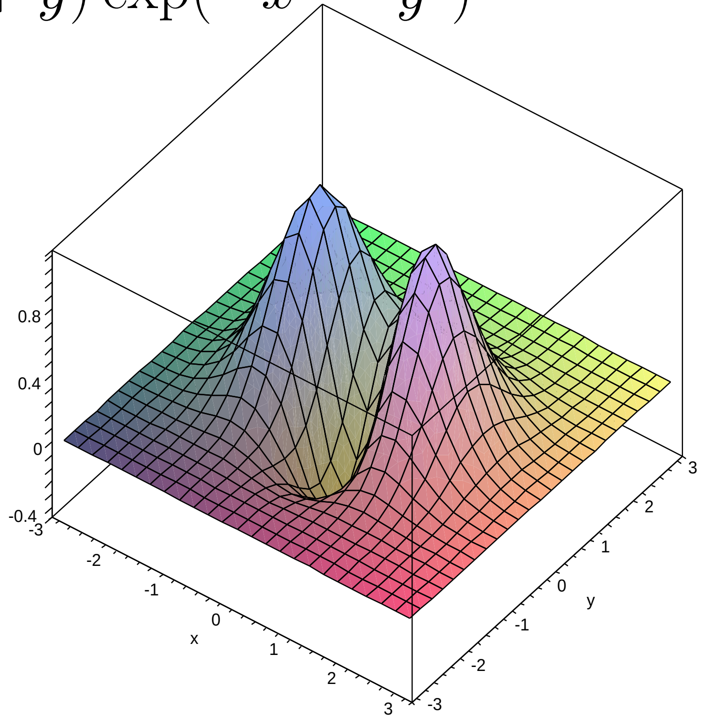
Steepest Descent Algorithm (cont.)

$$\begin{aligned} E(\mathbf{w}^{(i+1)}) &\approx E(\mathbf{w}^{(i)}) + (\nabla E(\mathbf{w}^{(i)}))^T \Delta \mathbf{w}^{(i)} \\ &= E(\mathbf{w}^{(i)}) - \eta \|\nabla E(\mathbf{w}^{(i)})\|^2 \end{aligned}$$

For positive learning rate η, $E(\mathbf{w}^{(i)})$ decreases in each iteration step (for small enough learning rates).

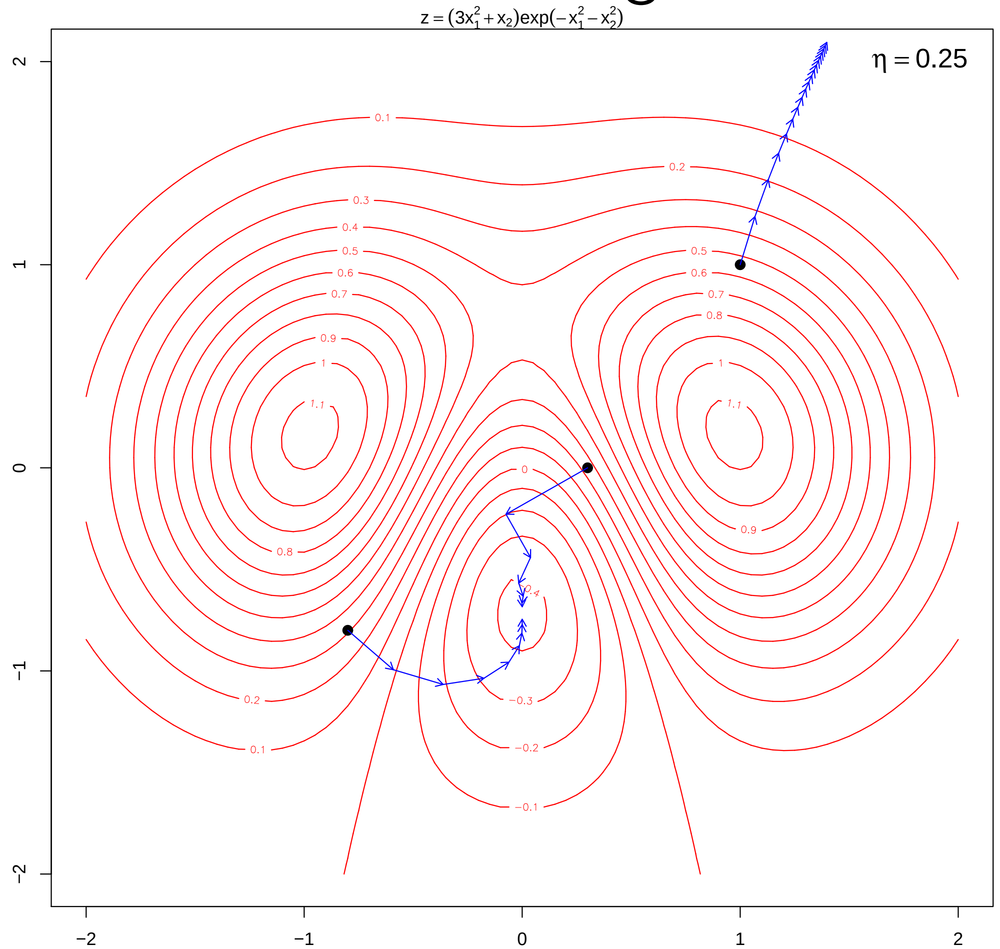
$(3x^2 + y)\exp(-x^2 - y^2)$





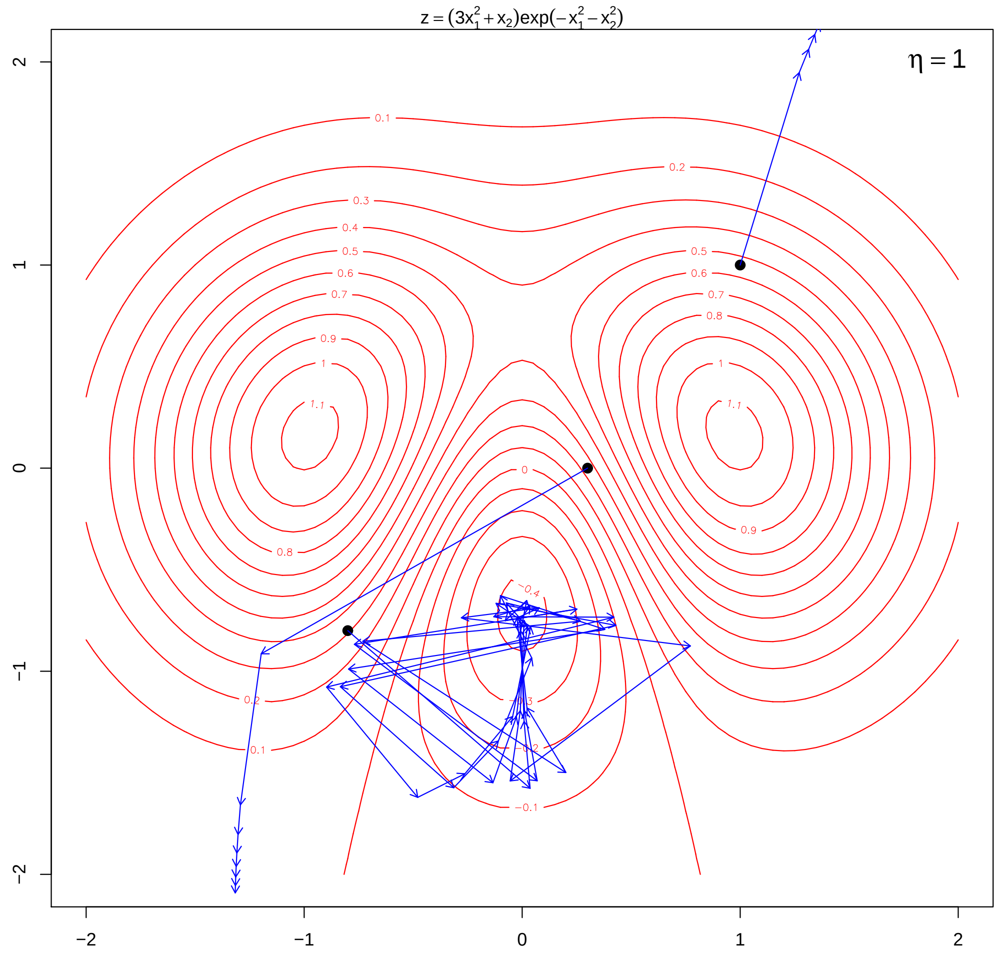
Steepest Descent Algorithm Example

Black points denote different starting values. Learning rate η is properly chosen, however for starting value $(1, 1)$, algorithm converges not to the global minimum. It follows steepest descent in the "wrong direction", in other words, gradient based algorithms are local search algorithms.



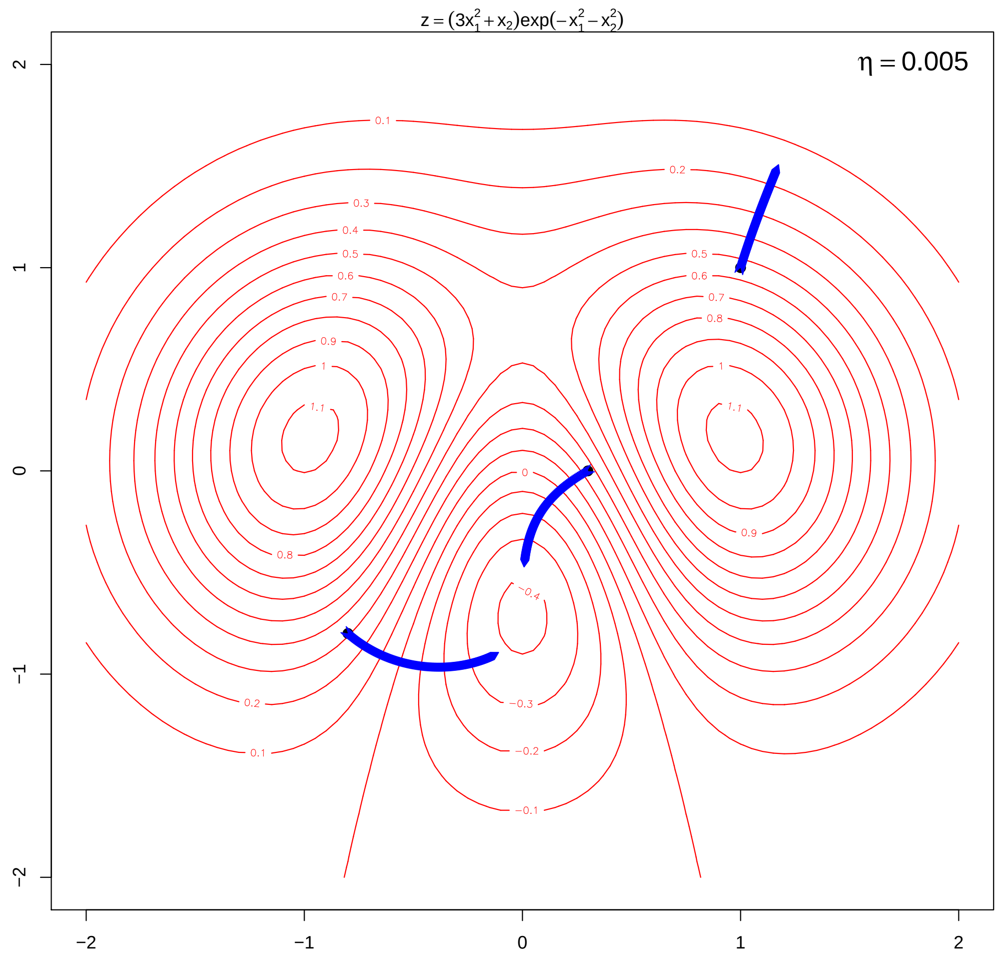
Steepest Descent Algorithm Example (cont.)

Learning rate $\eta = 1.0$ is too large, algorithm oscillates in a "*zig-zag*" manner or "*overleap*" the global minimum.

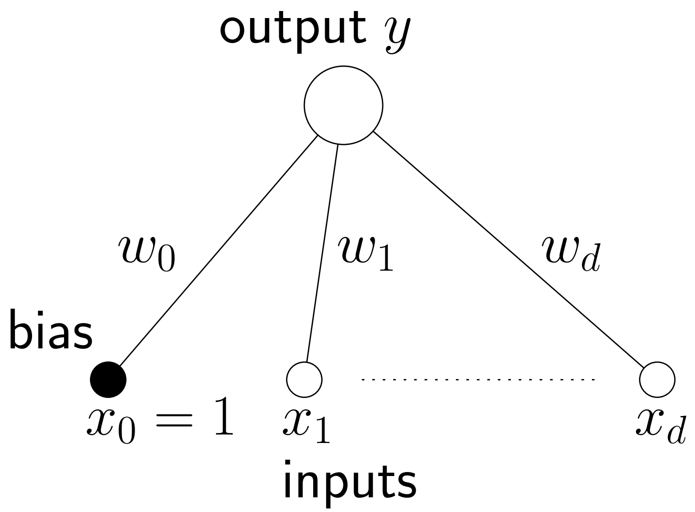


Steepest Descent Algorithm Example (cont.)

Learning rate $\eta = 0.005$ is too small, algorithm converges "very slowly".



Single-Layer Network



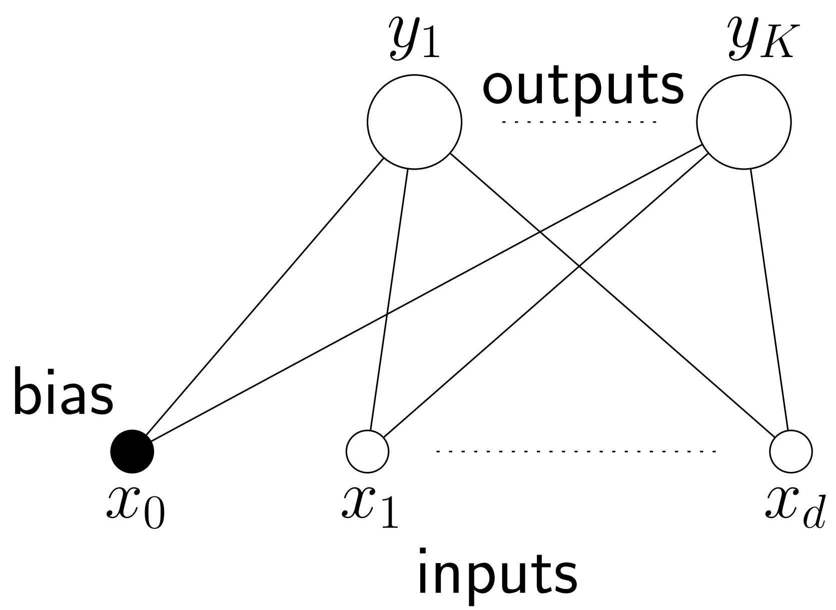
Equivalent notation:

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^T \widetilde{\mathbf{x}} = \sum_{i=0}^{d} w_i x_i$$

where $\widetilde{\mathbf{w}} = (w_0, \mathbf{w})$ and $\widetilde{\mathbf{x}} = (1, \mathbf{x})$.

Single-Layer Network (Multiple classes)

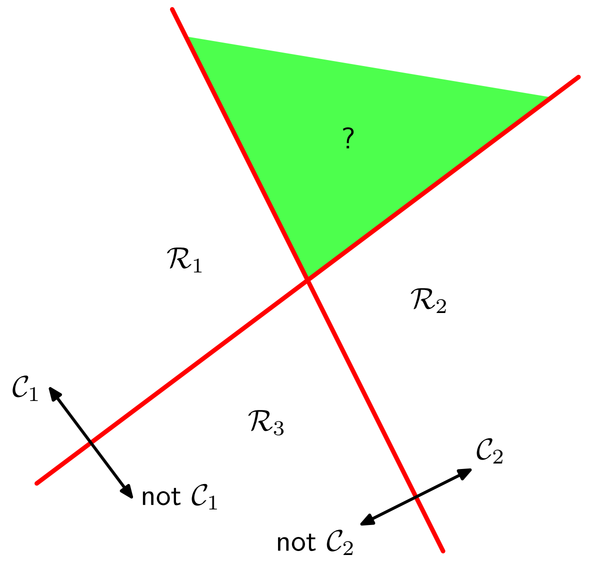
- Extension of linear discrimination functions to $K > 2$ classes.
- K-class discrimination function by combining a number of two-class discrimination functions.



Multiple classes (cont.)

One-versus-the-rest:
$K - 1$ classifiers each of which solves a two-class problem of separating points in a particular class $\mathcal{C}_k$ from points *not* in that class.

Leads to regions that are ambiguously classified.

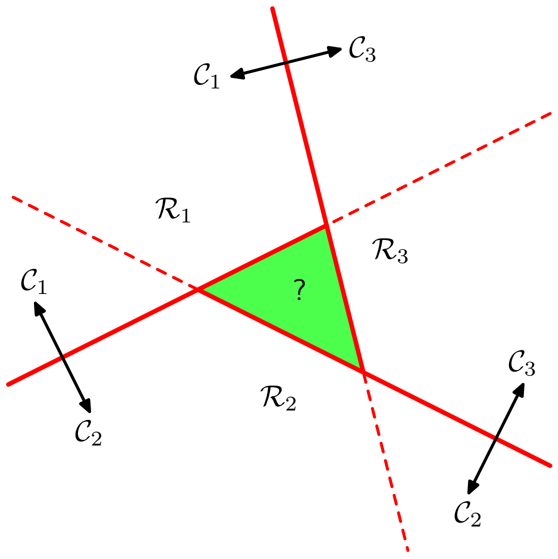


Multiple classes (cont.)

One-versus-one:

- $K(K-1)/2$ discriminant functions, one for every possible pair of classes.
- Example: $K=3$, $\texttt{discriminate}(\mathcal{C}_1, \mathcal{C}_2)$, $\texttt{discriminate}(\mathcal{C}_1, \mathcal{C}_3)$, $\texttt{discriminate}(\mathcal{C}_2, \mathcal{C}_3)$.

Leads again to regions that are ambiguously classified.



Multiple classes (cont.)

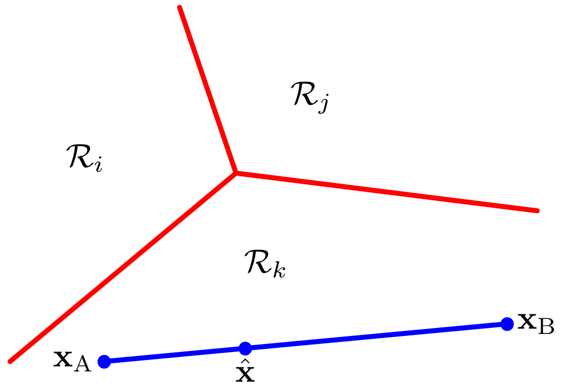
Avoid problem of ambiguous decision regions:

K-class discrimination function

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} = \sum_{i=1}^{d} w_{ki} x_i + w_{k0}$$

assign $\mathbf{x}$ to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$.

Such a discrimination function leads to decision regions that are always singly connected and convex.



All points of the form $\widehat{\mathbf{x}} = \alpha \mathbf{x}_A + (1-\alpha)\mathbf{x}_B$, that is, on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ lie in $\mathcal{R}_k \Rightarrow$ singly connected and convex.

Activation functions

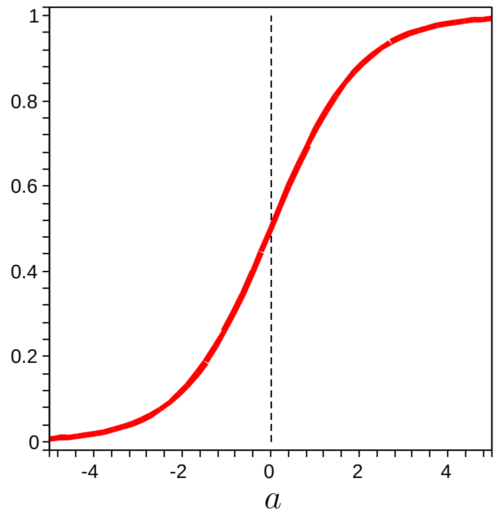
Discrimination functions of the form $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ are simple linear functions of the input variables $\mathbf{x}$, where distances are measured by means of the dot product.

Let us consider the non-linear *logistic sigmoid* activation function $g(\cdot)$ for limiting the output to $(0, 1)$, that is,

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

where

$$g(a) = \frac{1}{1 + \exp(-a)}$$

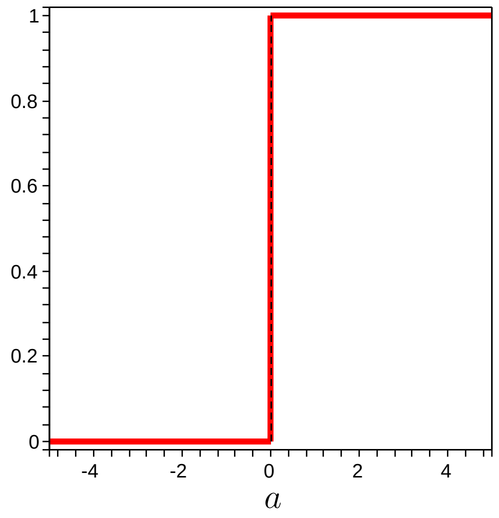


Single-layer network with a logistic sigmoid activation function can also output posterior probabilities (rather than geometric distances).

Activation functions (cont.)

Heaviside step function:

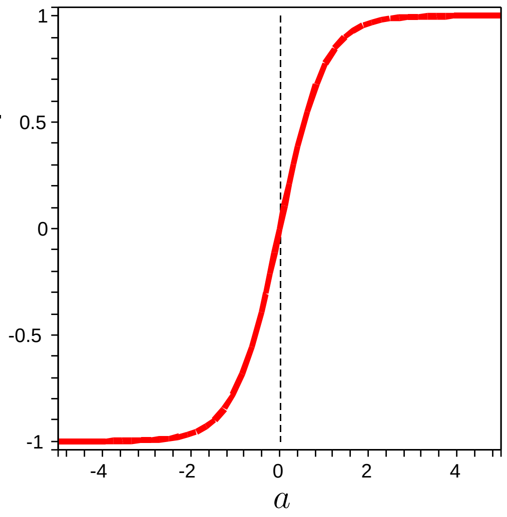
$$g(a) = \begin{cases} 0 & \text{if } a < 0 \\ 1 & \text{if } a \geq 0 \end{cases}$$



Hyperbolic tangent function:

$$g(a) = \tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$$

Note, $\tanh(a) \in (-1, 1)$



1
0.5
0
-0.5
-1
-4
-2
0
2
4
a