UCB - CS189
Introduction to Machine Learning
Fall 2015
Lecture 4: Learning as Risk Minimization

Isabelle Guyon

ChaLearn

Come to my office hours…

Wed 2:30-4:30 Soda 329

Last time

Kernel "Trick"

- $f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x})$
- $k(\mathbf{x}^k, \mathbf{x}) = \Phi(\mathbf{x}^k) \bullet \Phi(\mathbf{x})$

NON PARAMETRIC

Dual forms

- $f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x})$
- $\mathbf{w} = \sum_k \alpha_k \, \Phi(\mathbf{x}^k)$

PARAMETRIC

Come to my office hours…
Wed 2:30-4:30 Soda 329

Today

How to Train?

- Define a risk functional R[f(**x**,**w**)]
- Find a method to optimize it, typically "gradient descent"

$$w_j \leftarrow w_j - \eta \, \partial R / \partial w_j$$

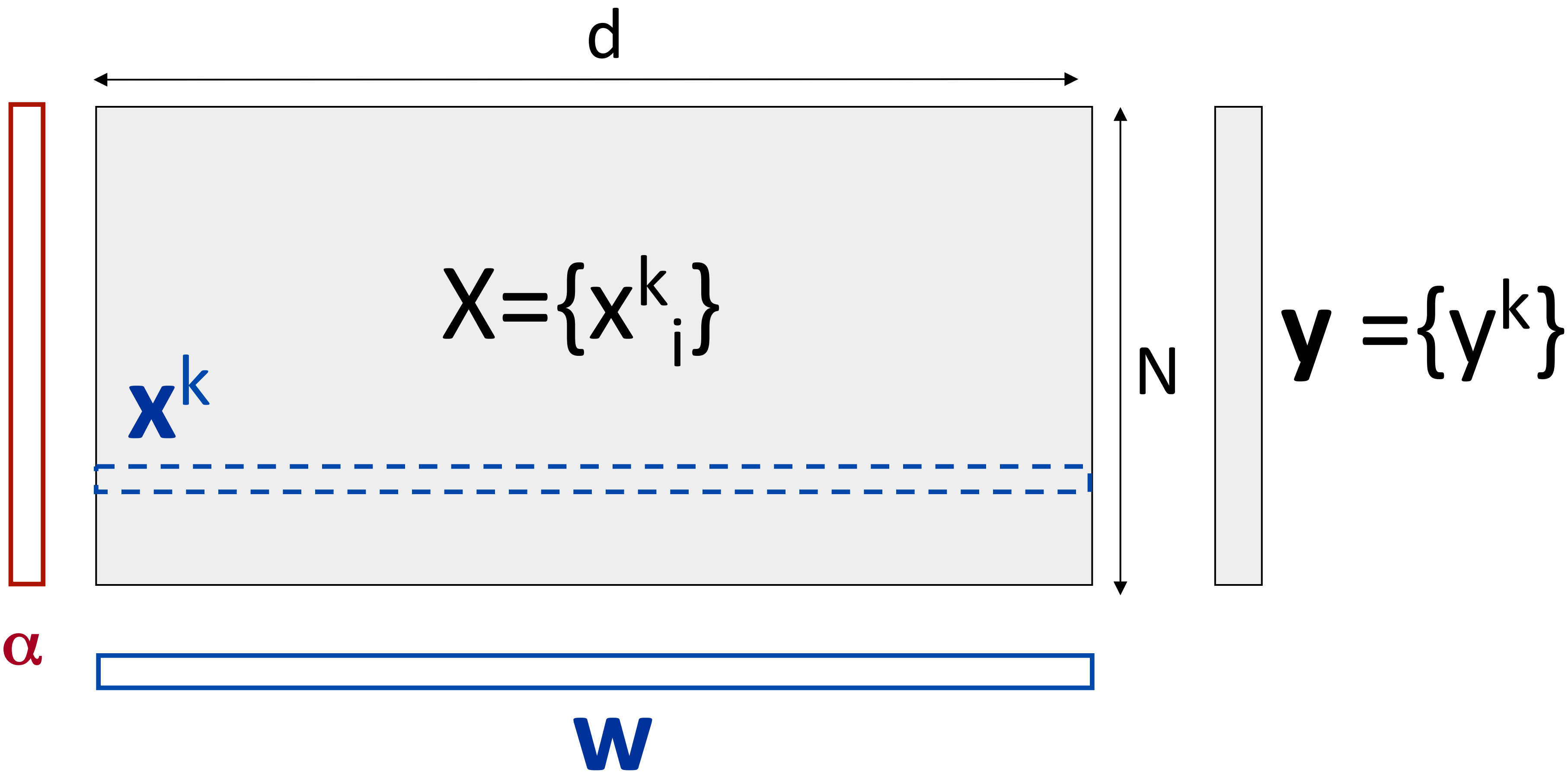
or any optimization method (mathematical programming, simulated annealing, genetic algorithms, etc.)

Machine Learning

(reminder)

- **Learning machines include**:
  - Linear discriminant
  - Kernel methods
  - Neural networks
- **Learning is tuning:**
  - Parameters (weights $\mathbf{w}$ or α, threshold b)
  - Hyperparameters (basis functions, kernels, number of units)

Conventions

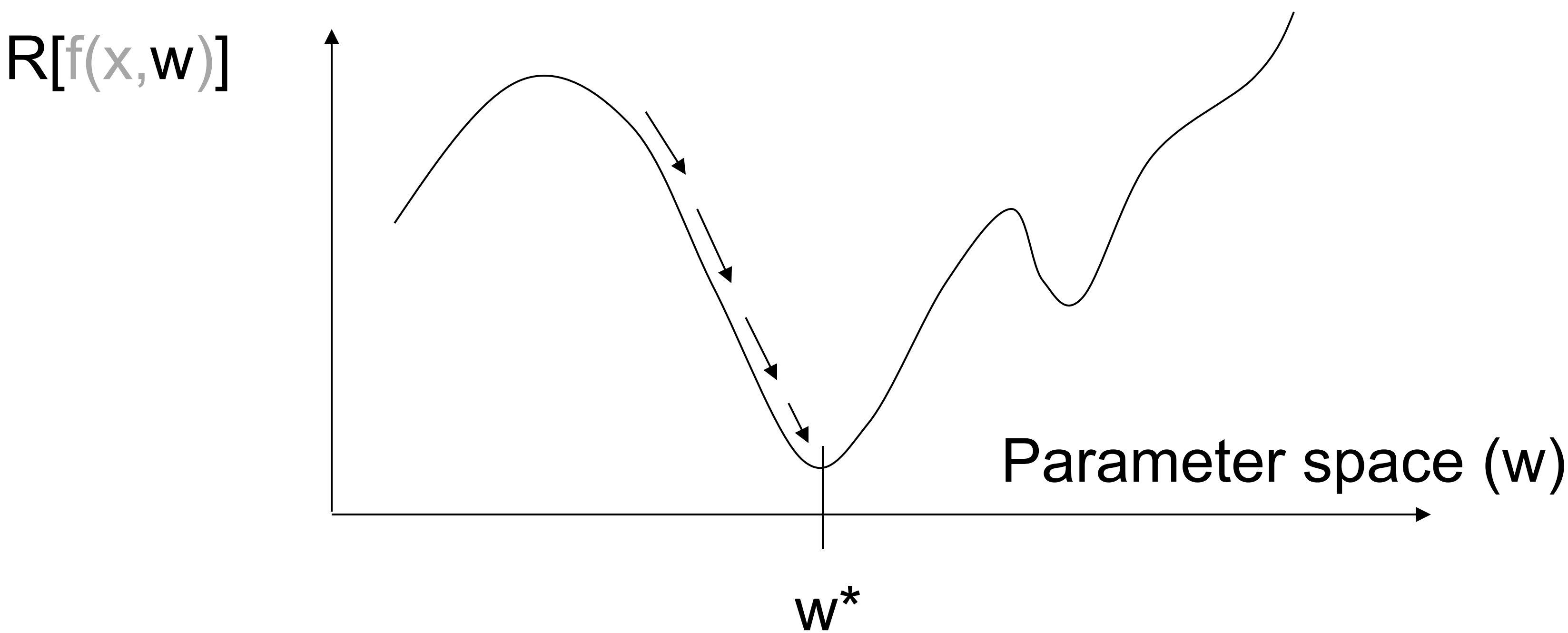


Math prerequisites

- Derivative and variation
- Slope and derivative
- Derivative chain rule
- Gradient
- Taylor series
- Hessian

What is a Risk Functional?

- A function of the parameters w of your learning machine f(x, w), assessing how much it is expected to fail on a given task.



Examples of risk functionals

Also called “objective functions” or “cost functions”

- **Classification:**
  - **Error rate:** $(1/N)\,\Sigma_{k=1:N}\,\underbrace{\mathbf{1}(\,\mathrm{sgn}(f(\mathbf{x}^k)) \neq y^k\,)}_{\text{0/1 loss}}$
- **Regression:**
  - **Mean square error:** $(1/N)\,\Sigma_{k=1:N}\,\underbrace{(\,f(\mathbf{x}^k) - y^k\,)^2}_{\text{square loss}}$

Examples of risk functionals

Also called "objective functions" or "cost functions"

- **Classification:**
  - **Error rate:** $(1/N)\, \Sigma_{k=1:N}\, \underbrace{\mathbf{1}(\ \mathrm{sgn}(f(\mathbf{x}^k)) \neq y^k\ )}_{\textbf{0/1 loss}}$
- **Regression:**
  - **Mean square error:** $(1/N)\, \Sigma_{k=1:N}\, \underbrace{(\ f(\mathbf{x}^k) - y^k\ )^2}_{\textbf{square loss}}$

How to Train?

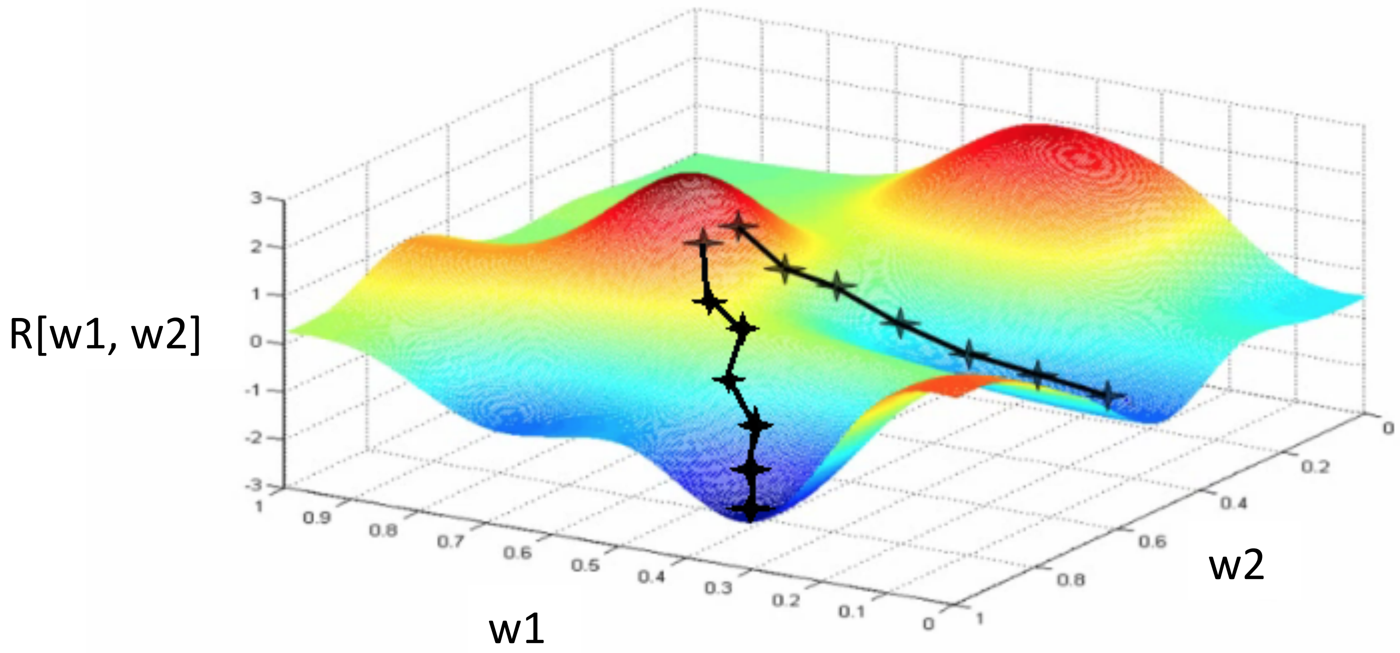
- Define a risk functional $R[f(\mathbf{x},\mathbf{w})]$
- Find a method to optimize it, typically "gradient descent"

$$w_i \leftarrow w_i - \eta\ \partial R/\partial w_i$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\ \nabla_{\mathbf{w}} R$$

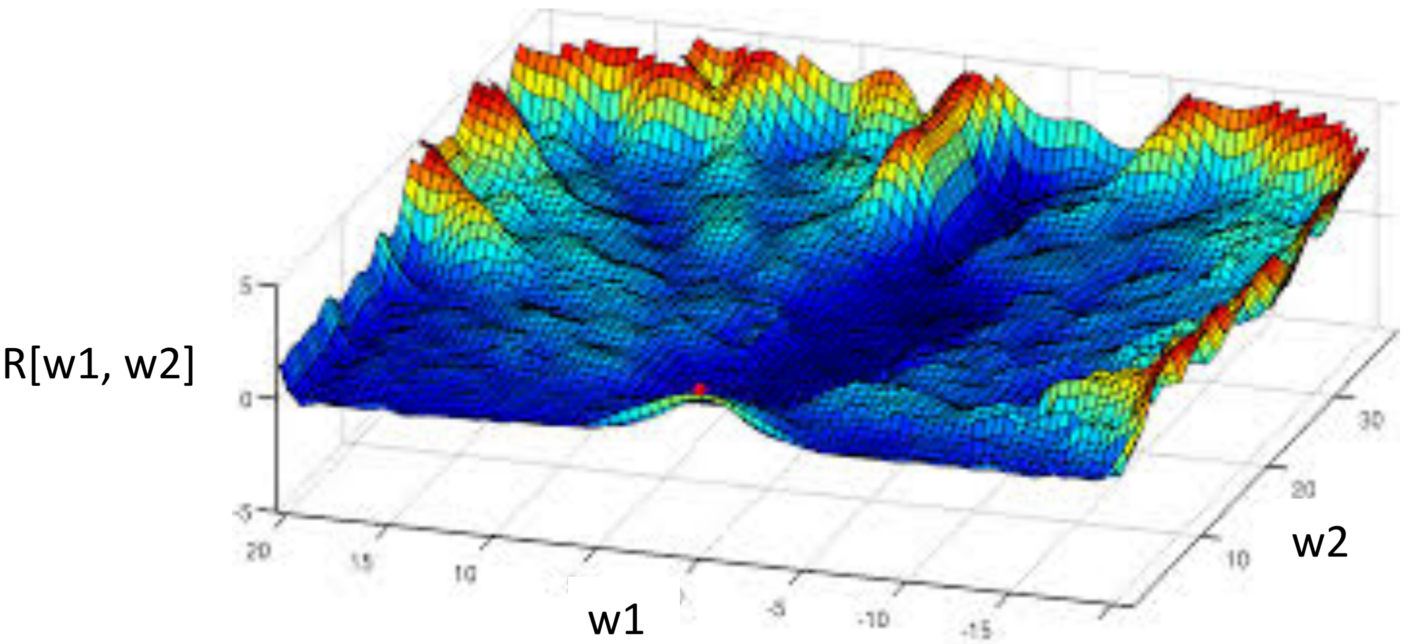
or any optimization method (mathematical programming, simulated annealing, genetic algorithms, etc.)

Gradient descent falls into local minima…

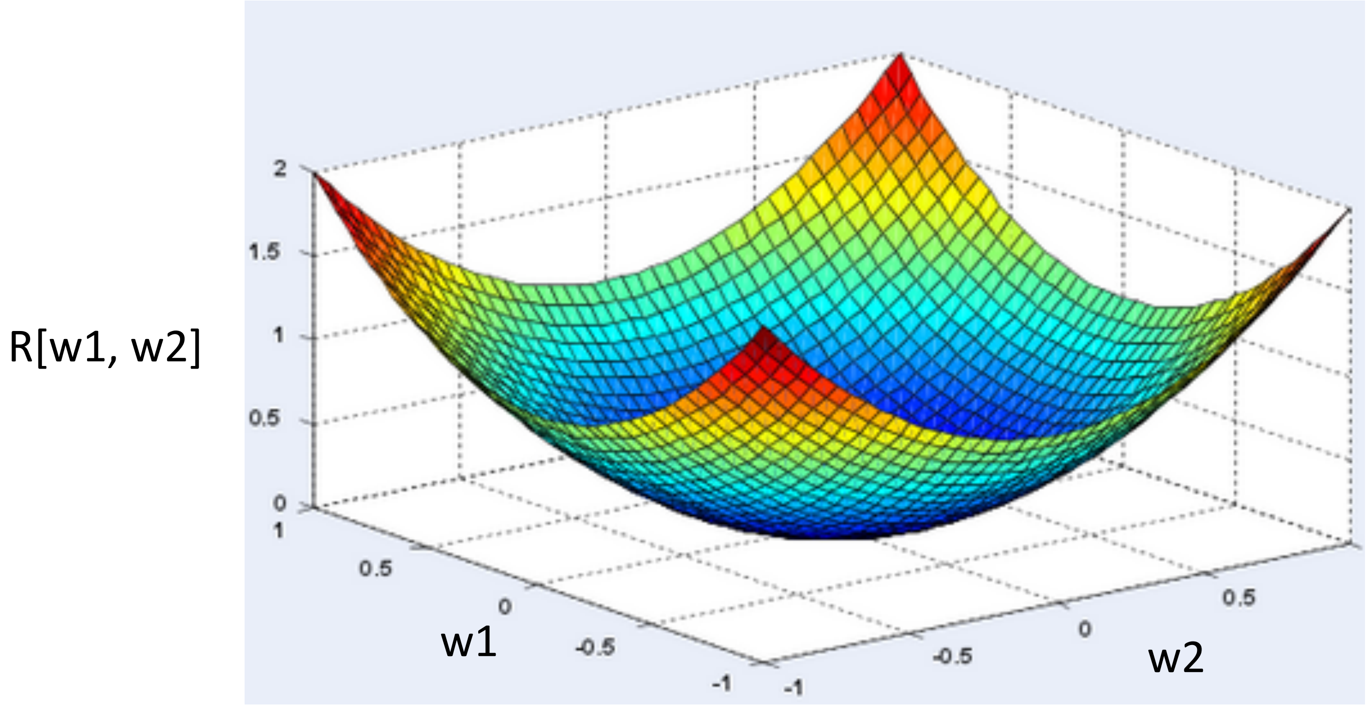


η is the learning rate of gradient step.

… finding the global optimum can be hard …



… except if the risk functional is convex!



Learning rate η

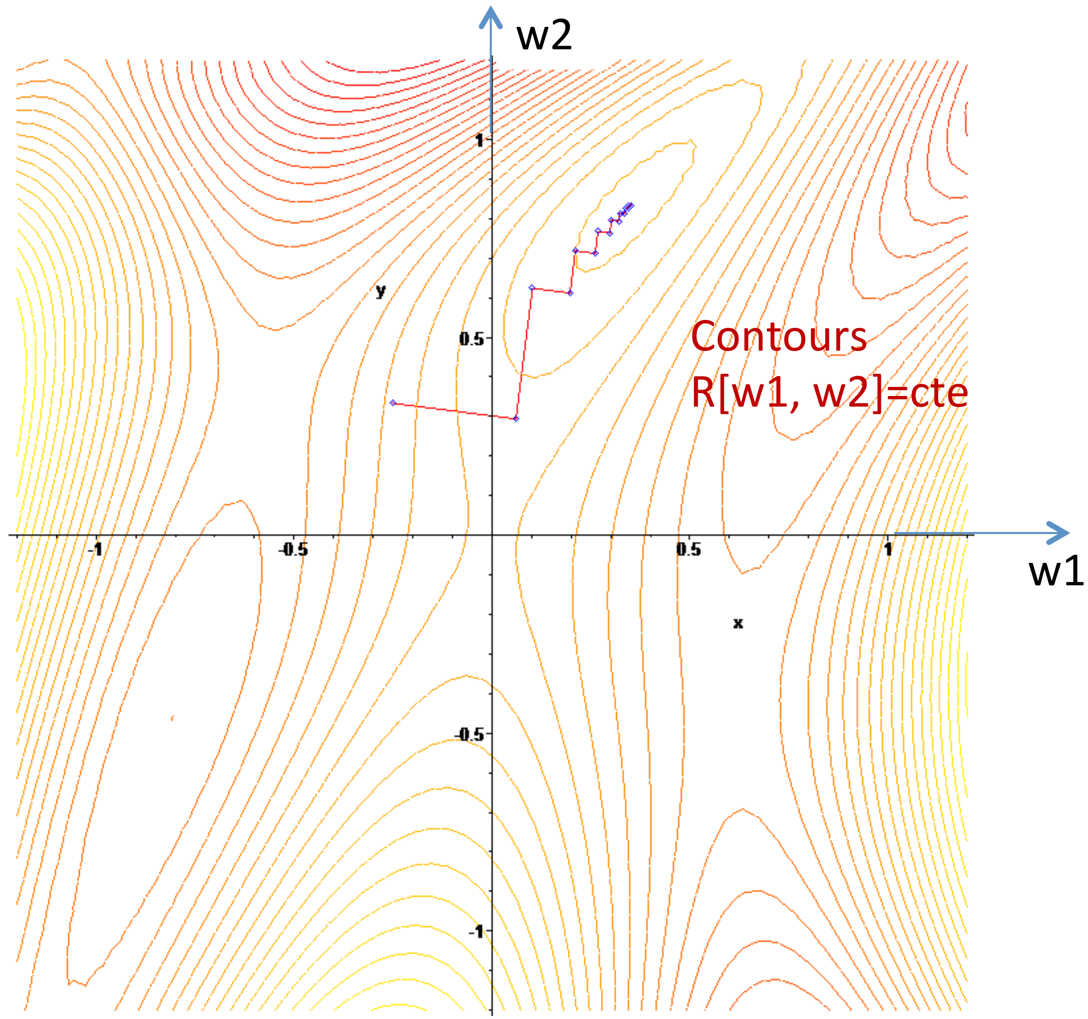
$\mathbf{w} \leftarrow \mathbf{w} - \eta\ \nabla_{\mathbf{w}} R$

η **too small:** many steps needed to converge.

η **too large:** zigzags.

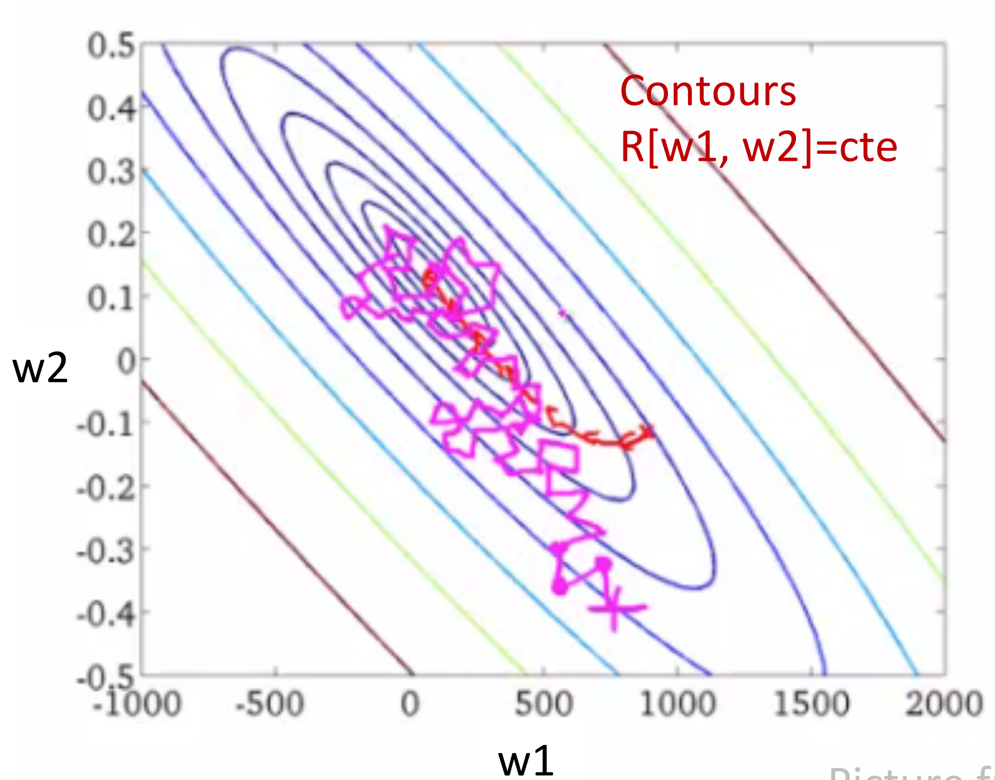
η **optimal** to second order:
$\mathbf{w} \leftarrow \mathbf{w} - H^{-1}\ \nabla_{\mathbf{w}} R$
Newton's method

$H = [\partial^2 R/\partial w_i \partial w_j\ ]$
Hessian



Picture from Wikipedia

Stochastic gradient



Picture from holehouse.org

Possible recipe (Bottou, 2010):

- Shuffle examples randomly
- Fix the learning rate (~t^{-a} 0.5≤a≤1; Xu, 2010, Bach, 2015; experiment on data subset)
- Average of w(t) to smooth the result (Polyak, Juditsky, 1992)
- One pass only for “big data”

The risk is the sum of "losses"

$$R[f] = (1/N) \Sigma_{k=1:N} L(f(\mathbf{x}^k), y^k)$$

- $L(f(\mathbf{x}), y) = \mathbf{1}(\mathrm{sgn}(f(\mathbf{x}))\neq y) = \mathbf{1}(yf(\mathbf{x})<0)$ zero-one loss
- $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2 = (yf(\mathbf{x}) - 1)^2$ square loss

with $y=\pm 1$

Losses are conveniently expressed as a function of the "functional margin" $z = y\,f(\mathbf{x})$

- $L(f(\mathbf{x}), y) = \mathbf{1}(z<0)$ zero-one loss
- $L(f(\mathbf{x}), y) = (z - 1)^2$ square loss

The risk is the sum of "losses"

$$R[f] = (1/N) \Sigma_{k=1:N} L(f(\mathbf{x}^k), y^k)$$

- $L(f(\mathbf{x}), y) = \mathbf{1}(\mathrm{sgn}(f(\mathbf{x}))\neq y) = \mathbf{1}(yf(\mathbf{x})<0)$ zero-one loss
- $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2 = (yf(\mathbf{x}) - 1)^2$ square loss

with $y=\pm1$

Losses are conveniently expressed as a function of the "functional margin" $z = y\, f(\mathbf{x})$

- $L(f(\mathbf{x}), y) = \mathbf{1}(z<0)$ zero-one loss
- $L(f(\mathbf{x}), y) = (z - 1)^2$ square loss

The risk is the sum of "losses"

$$R[f] = (1/N)\,\Sigma_{k=1:N}\, L(\ f(\mathbf{x}^k),\ y^k\ )$$

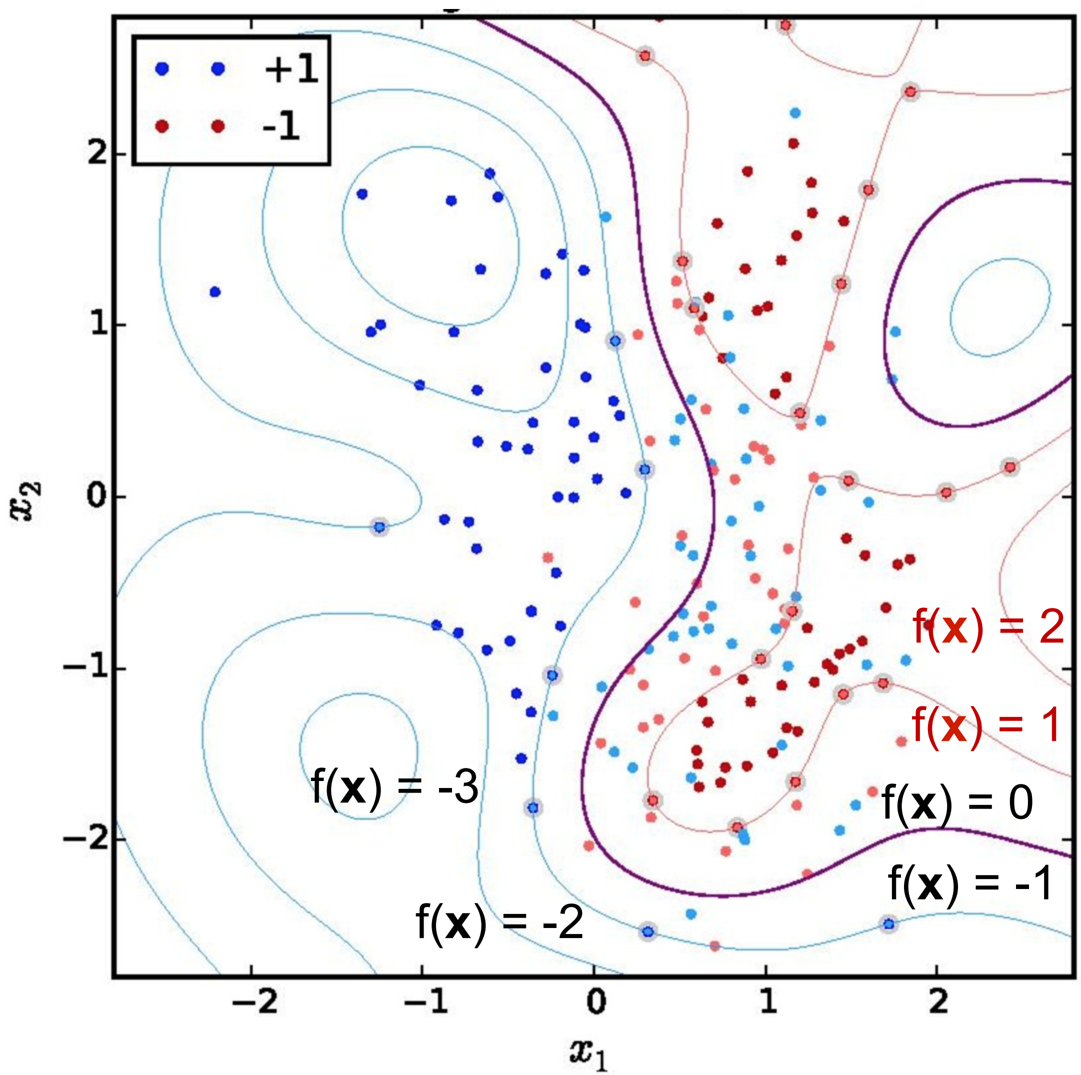
- $L(\ f(\mathbf{x}), y\ ) = \mathbf{1}(\mathrm{sgn}(f(\mathbf{x}))\neq y) = \mathbf{1}(\ yf(\mathbf{x})<0\ )$ zero-one loss
- $L(\ f(\mathbf{x}), y\ ) = (\ f(\mathbf{x}) - y\ )^2 = (\ yf(\mathbf{x}) - 1\ )^2$ square loss

with $y=\pm 1$

Losses are conveniently expressed as a function of the "funtional margin" $z = y\,f(\mathbf{x})$

- $L(\ f(\mathbf{x}), y\ ) = \mathbf{1}(\ z<0\ )$ zero-one loss
- $L(\ f(\mathbf{x}), y\ ) = (\ z - 1\ )^2$ square loss

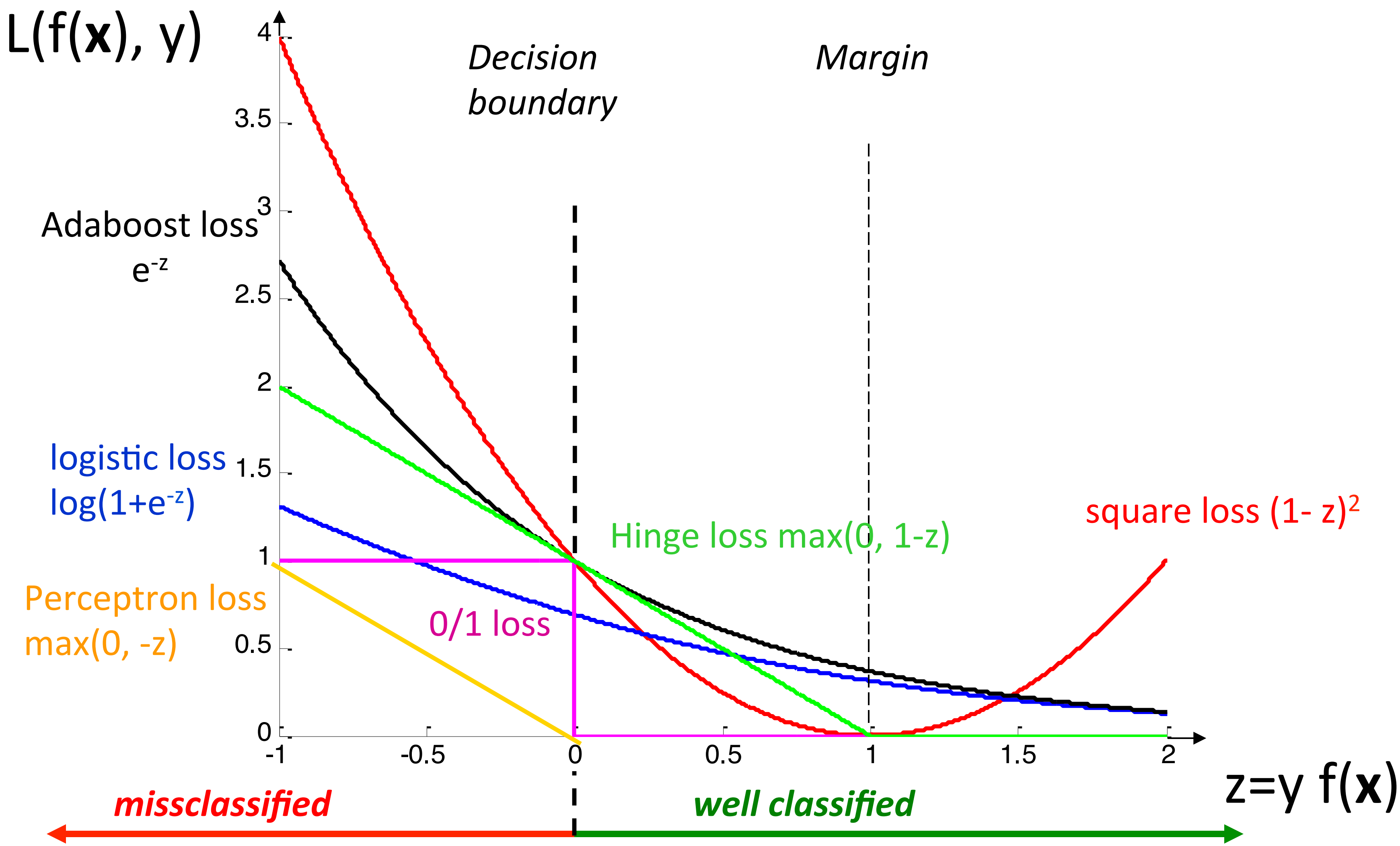
"Functional margin" z = y f(**x**)



Picture from Wikipedia

Loss Functions

The risk is the average of the loss.



Dual learning machines

PARAMETRIC

$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x})$

$\mathbf{w} = \sum_k \alpha_k \Phi(\mathbf{x}^k)$

Hebb's rule

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$

(Hebb 1949)

NON PARAMETRIC

$f(\mathbf{x}) = \sum_k \alpha_k k(\mathbf{x}^k, x)$

$k(\mathbf{x}^k, \mathbf{x}) = \Phi(\mathbf{x}^k).\Phi(\mathbf{x})$

Dual Hebb's rule

$\alpha_k \leftarrow \alpha_k + y_k$

Perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$ if $y_k f(\mathbf{x}^k)<0$

(Rosenblatt 1958)

Minover (optimum margin)

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$ for min $y^k f(\mathbf{x}^k)$

(Krauth-Mézard 1987)

Potential Function algorithm

$\alpha_k \leftarrow \alpha_k + y_k$ if $y_k f(\mathbf{x}^k)<0$

(Aizerman et al 1964)

Dual minover

$\alpha_k \leftarrow \alpha_k + y^k$ for min $y_k f(\mathbf{x}^k)$

(ancestor of SVM)

Dual learning machines

PARAMETRIC

$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x})$

$\mathbf{w} = \sum_k \alpha_k \Phi(\mathbf{x}^k)$

Perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$ if $y_k f(\mathbf{x}^k) < 0$

(Rosenblatt 1958)

Minover (optimum margin)

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$ for min $y^k f(\mathbf{x}^k)$

(Krauth-Mézard 1987)

LMS regression

$\mathbf{w} \leftarrow \mathbf{w} + \eta\, (y_k - f(\mathbf{x}_k))\, \Phi(\mathbf{x}^k)$

NON PARAMETRIC

$f(\mathbf{x}) = \sum_k \alpha_k\, k(\mathbf{x}^k, x)$

$k(\mathbf{x}^k, \mathbf{x}) = \Phi(\mathbf{x}^k).\Phi(\mathbf{x})$

Potential Function algorithm

$\alpha_k \leftarrow \alpha_k + y_k$ if $y_k f(\mathbf{x}^k) < 0$

(Aizerman et al 1964)

Dual minover

$\alpha_k \leftarrow \alpha_k + y^k$ for min $y_k f(\mathbf{x}^k)$

(ancestor of SVM 1992, similar to kernel Adatron, 1998, and SMO, 1999)

Dual LMS

$\alpha_i \leftarrow \alpha_i + \eta\, (y_i - f(\mathbf{x}^k))$

Exercise: Gradient Descent

- Linear discriminant $f(\mathbf{x}) = \Sigma_i w_i x_i$
- Functional margin $z = y f(\mathbf{x})$, $y=\pm 1$
- Compute $\partial z/\partial w_i$
- Derive the learning rules $\Delta w_i = -\eta\, \partial L/\partial w_i$ corresponding to the following loss functions:

square loss
$L=(1- z)^2$

SVC loss
$L=\max(0, 1-z)$

Adaboost loss
$L=e^{-z}$

Perceptron loss
$L=\max(0, -z)$

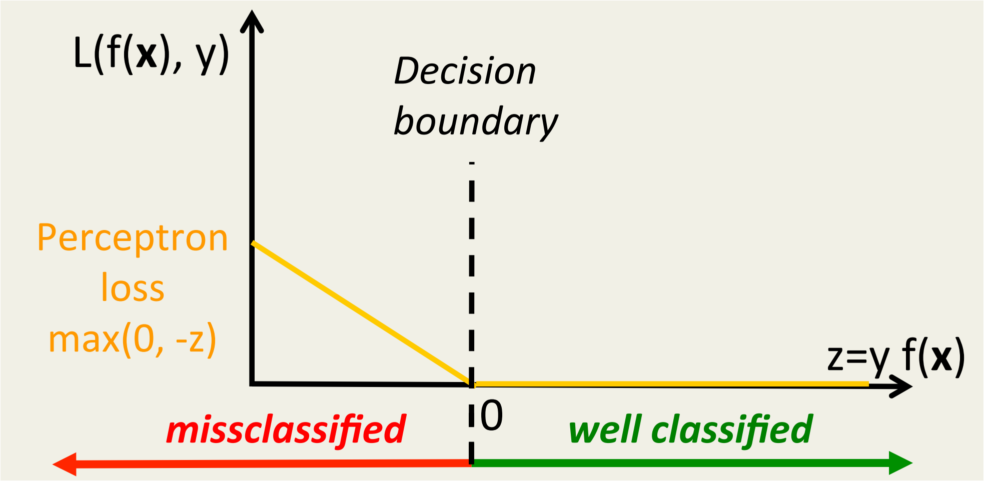
logistic loss
$L=\log(1+e^{-z})$

Example: the Perceptron algorithm

Rosenblatt, 1957

- $f(\mathbf{x}) = \Sigma_i \, w_i \, x_i$
- $z = y \, f(\mathbf{x}) = \Sigma_i \, w_i \, y \, x_i$
- $\partial z / \partial w_i = y \, x_i$
- $L_{perceptron} = \max(0, -z)$
- $\Delta w_i = -\eta \, \partial L / \partial w_i$

$$= -\eta \, \partial L / \partial z \, . \, \partial z / \partial w_i$$



$$\boxed{\Delta w_i = \begin{cases} \eta \, y \, x_i, & \text{if } z<0 \text{ (misclassified example)} \\ 0 & \text{otherwise} \end{cases}}$$

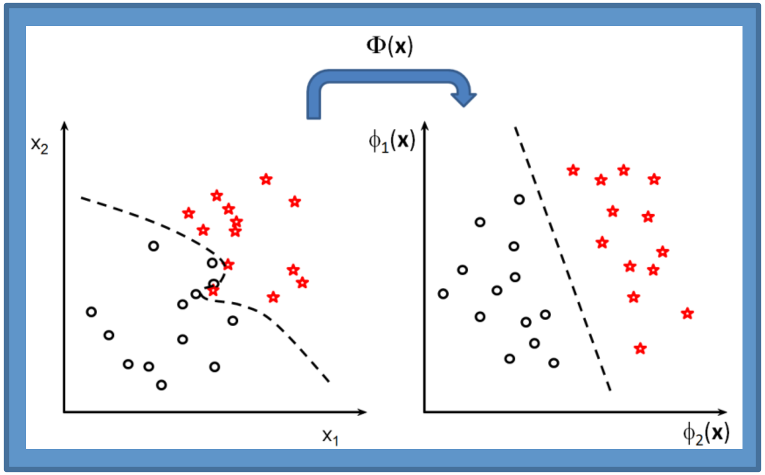
Like Hebb's rule but for misclassified examples only.

Example: the Perceptron algorithm

Rosenblatt, 1957

- $f(\mathbf{x}) = \sum_i w_i \, \Phi_i(\mathbf{x})$
- $z = y\, f(\mathbf{x}) = \sum_i w_i \, y \, \Phi_i(\mathbf{x})$
- $\partial z / \partial w_i = y \, \Phi_i(\mathbf{x})$
- $L_{perceptron} = \max(0, -z)$
- $\Delta w_i = -\eta \, \partial L / \partial w_i$

  $= -\eta \, \partial L / \partial z \, . \, \partial z / \partial w_i$



$$\Delta w_i = \begin{cases} \eta \, y \, \Phi_i(\mathbf{x}), \text{ if } z<0 \text{ (misclassified example)} \\ 0 \ \ \text{otherwise} \end{cases}$$

Like Hebb's rule but for misclassified examples only.

Dual algorithm: Potential function learning algorithm

Aizerman, Braverman, Rozonoer, 1964

- Perceptron: $\Delta w_i = \eta\ y\ x_i$, if $z<0$, 0 otherwise
- For example $\mathbf{x}^k$:

$$\Delta \mathbf{w} = \eta\ y^k\ \mathbf{x}^k, \text{ if } z<0, 0 \text{ otherwise}$$

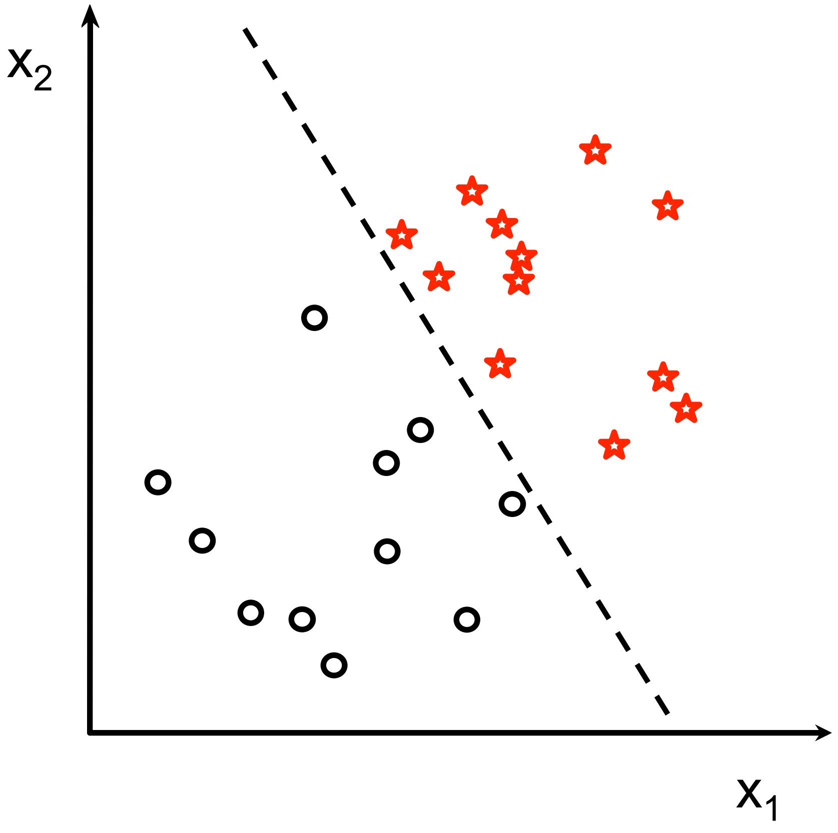
- $\mathbf{w} = \Sigma_k\ \alpha_k\ \mathbf{x}^k$, $\quad \Delta \mathbf{w} = \Delta\alpha_k\ \mathbf{x}^k$

$$\boxed{\Delta\alpha_k = \eta\ y^k}$$

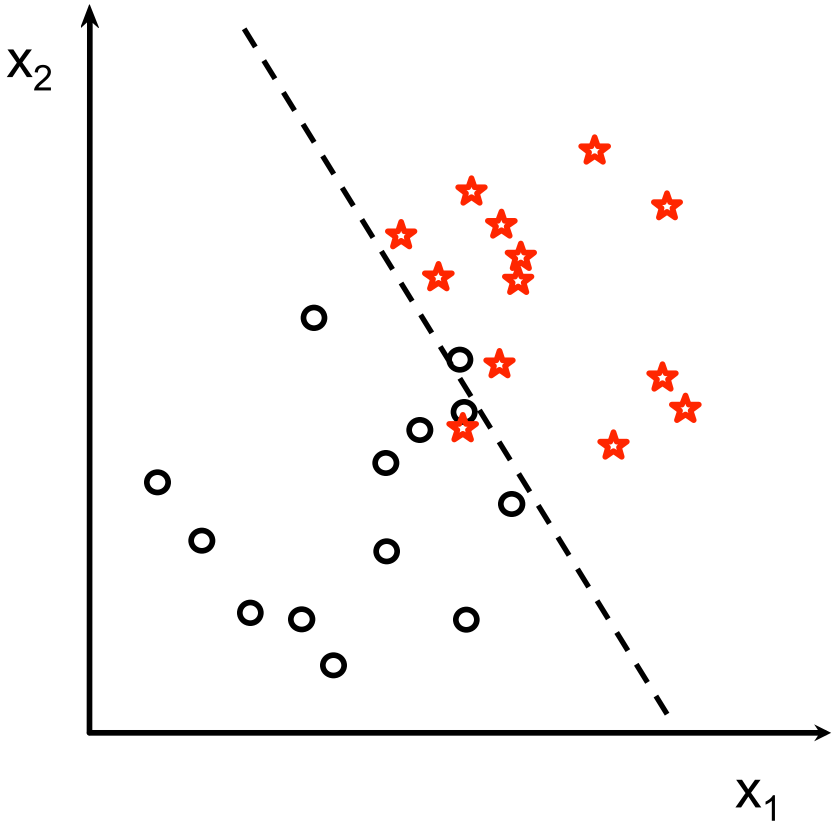
Note: the $\Delta\alpha$ is different when $\partial L/\partial\alpha$ is computed directly.

Linearly separable?

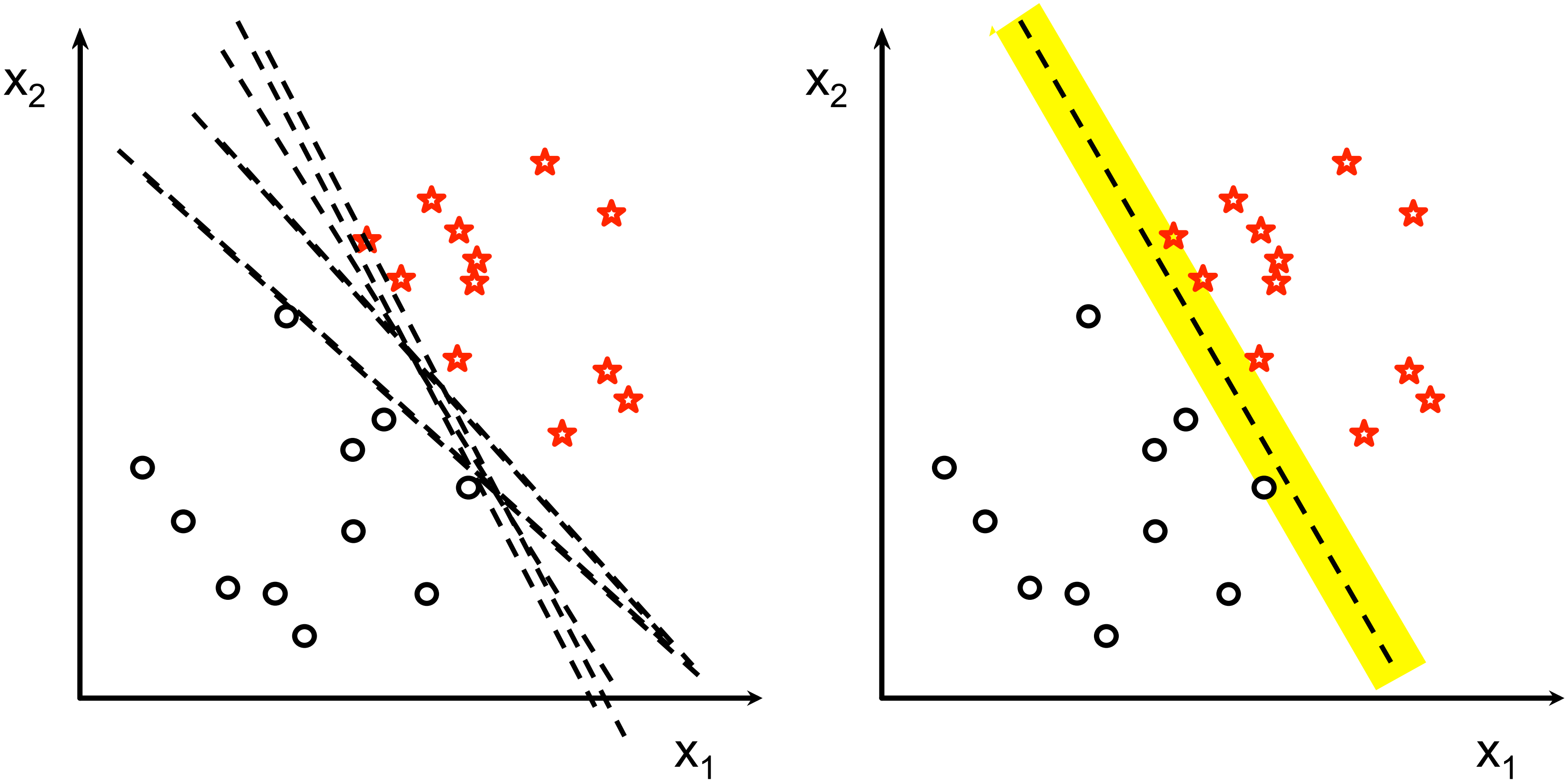
Yes



No

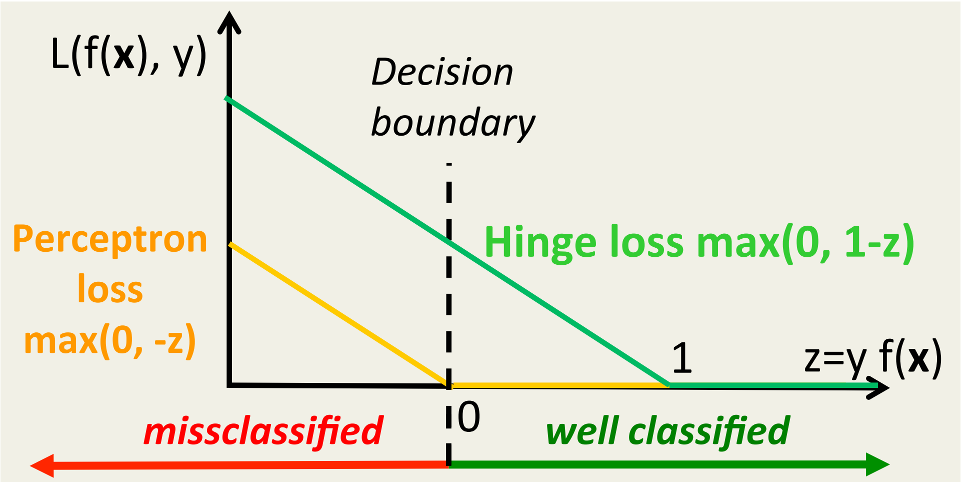


Large margin



Large margin Perceptron

- $L_{perceptron} = \max(0, -z)$
- $L_{hinge} = \max(0, 1 - z)$

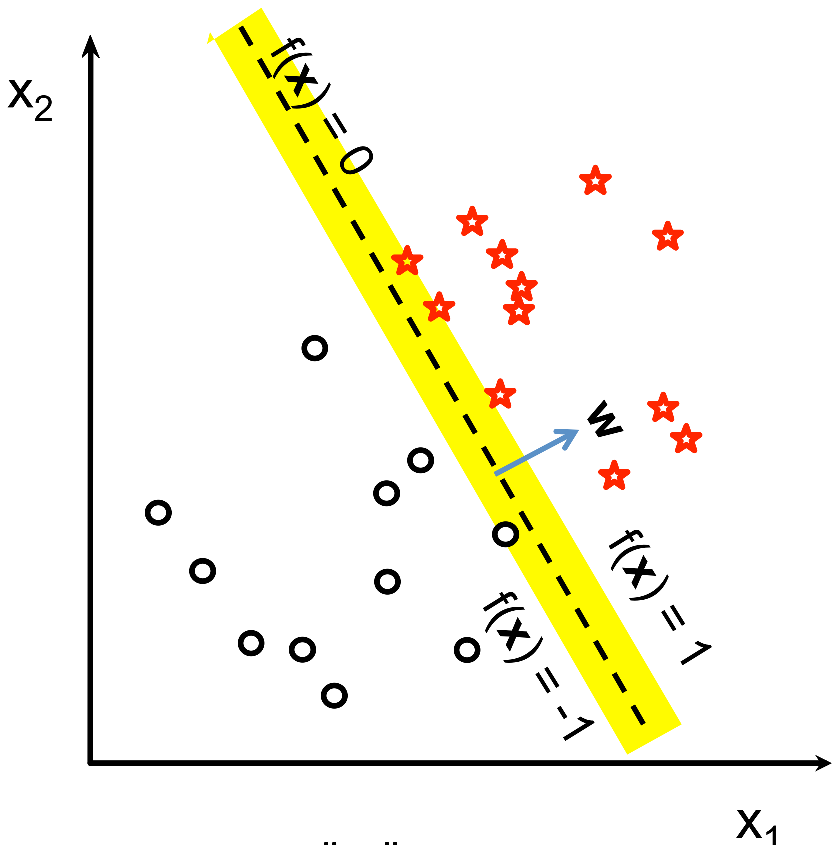


$$\Delta w_i = \begin{cases} \eta \quad y\, x_i, \text{ if } z < 1 \text{ (misclassified or within margin)} \\ 0 \quad \text{otherwise} \end{cases}$$

w must be normalized to give the scale!

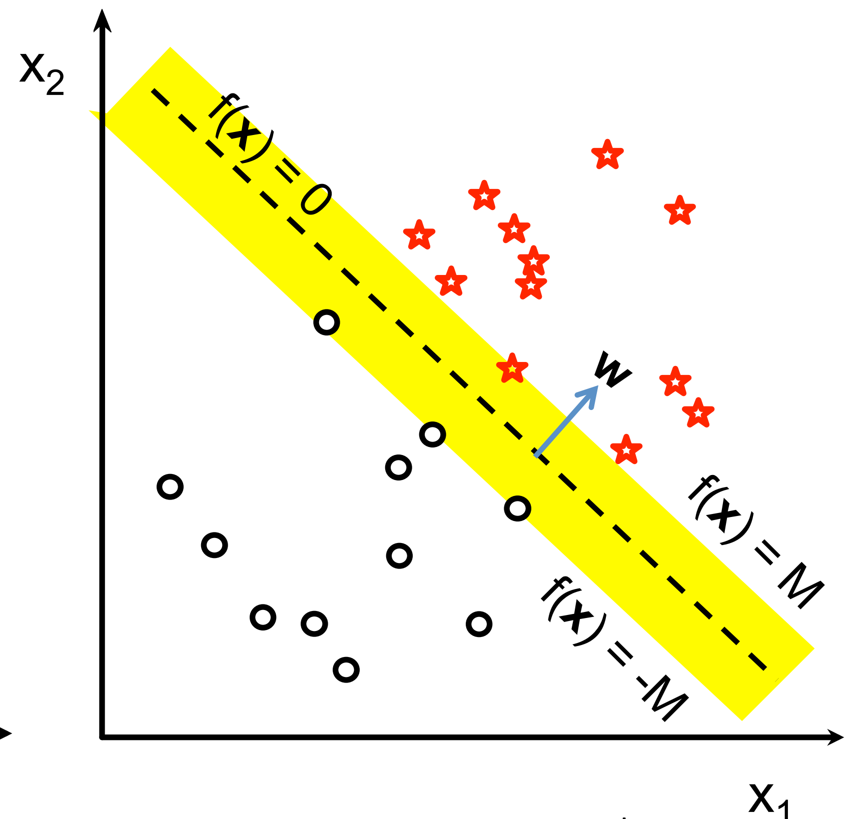
Optimum margin

Large margin



$\|\mathbf{w}\| = 1$

Optimum margin

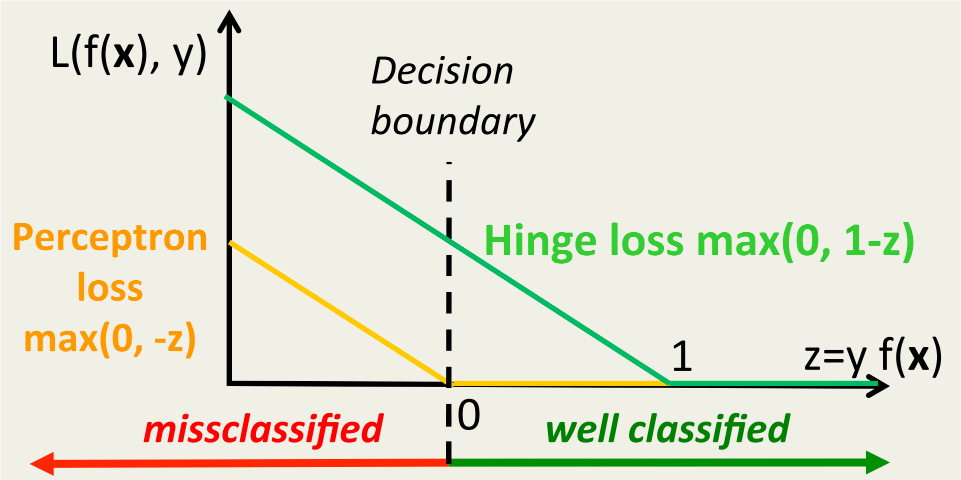


$M = \max_w (\min_k(f(\mathbf{x}^k)))$

Optimum margin Perceptron

(Minover, Krauth-Mézard 1987)

- $L_{perceptron} = \max(0, -z)$
- $L_{hinge} = \max(0, 1 - z)$



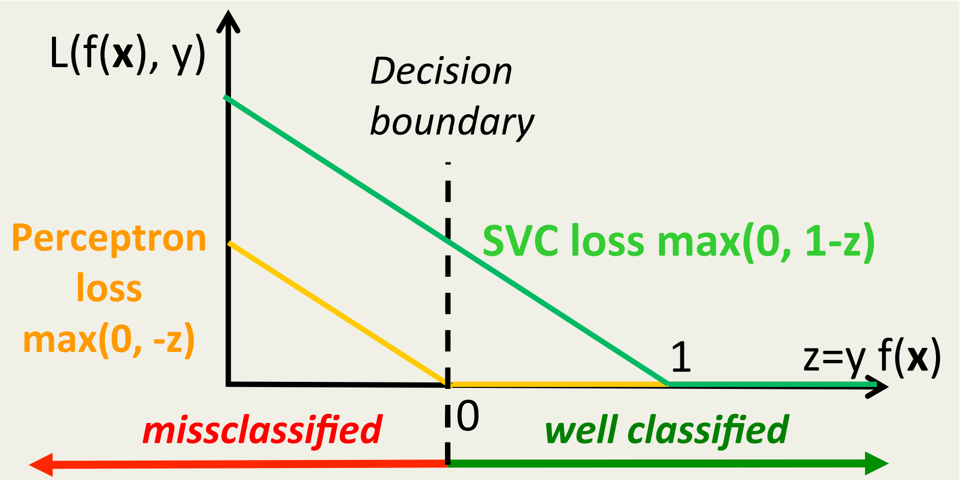
$$\Delta w_i \begin{cases} \eta \;\; y\, x_i, \text{ only for } \min(z) \\ 0 \;\; \text{otherwise} \end{cases} \qquad\qquad z = y\, f(\mathbf{x})$$

w must be normalized to give the scale!

Optimum margin Perceptron

(Minover, Krauth-Mézard 1987)

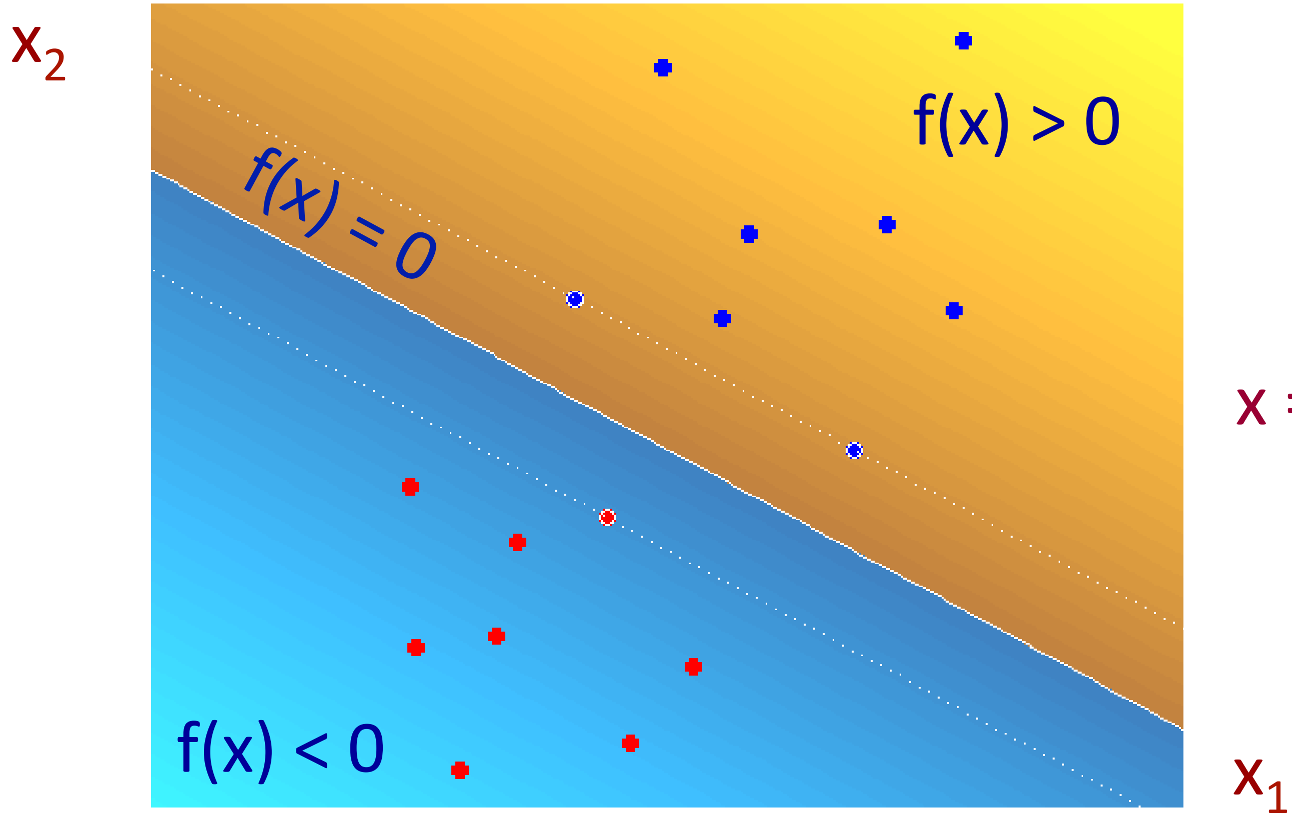
- $L_{perceptron} = \max(0, -z)$
- $L_{svc} = \max(0, 1 - z)$



$$\Delta w_i = \begin{cases} \eta \ \ y\, x_i, \text{ for } \min(z) \\ 0 \ \ \text{otherwise} \end{cases} \qquad \Delta \alpha_k = \begin{cases} \eta\, y^k, \text{ for } \min(z) \\ 0 \ \ \text{otherwise} \end{cases}$$

w must be normalized to give the scale!

Linear optimum margin classifier



$x = [x_1, x_2]$

Vapnik, 1962

$$f(x) = \Sigma_i \, w_i \, x_i + b$$

Kernel “Trick”

- $f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x})$
- $k(\mathbf{x}^k, \mathbf{x}) = \mathbf{\Phi}(\mathbf{x}^k) \bullet \mathbf{\Phi}(\mathbf{x})$

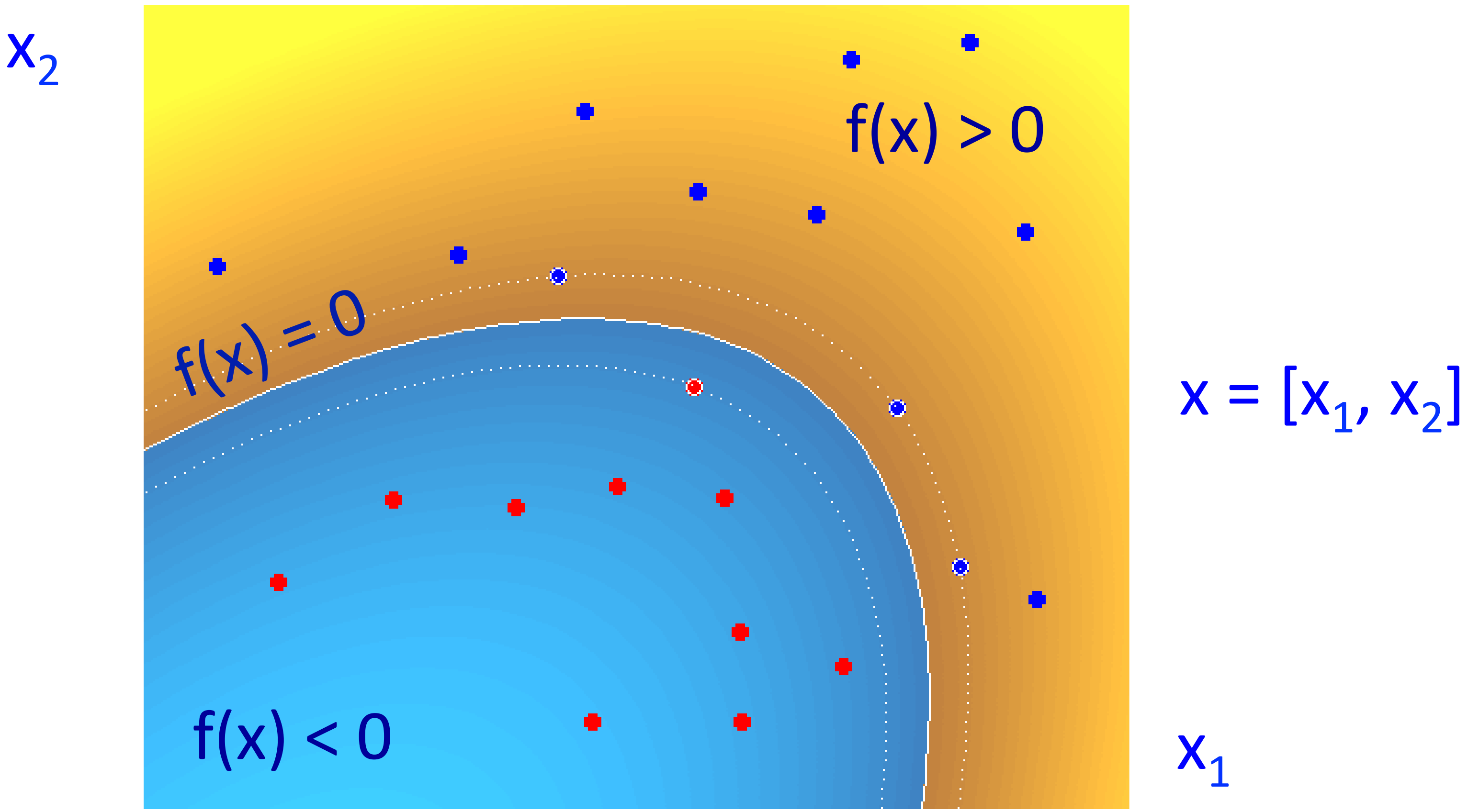
$\Delta\alpha_k = \eta \, y^k$, for min(z),
0 otherwise

Dual forms

- $f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{\Phi}(\mathbf{x})$
- $\mathbf{w} = \sum_k \alpha_k \, \mathbf{\Phi}(\mathbf{x}^k)$

$\Delta w = \eta \, y \, \mathbf{\Phi}(\mathbf{x})$, for min(z),
0 otherwise

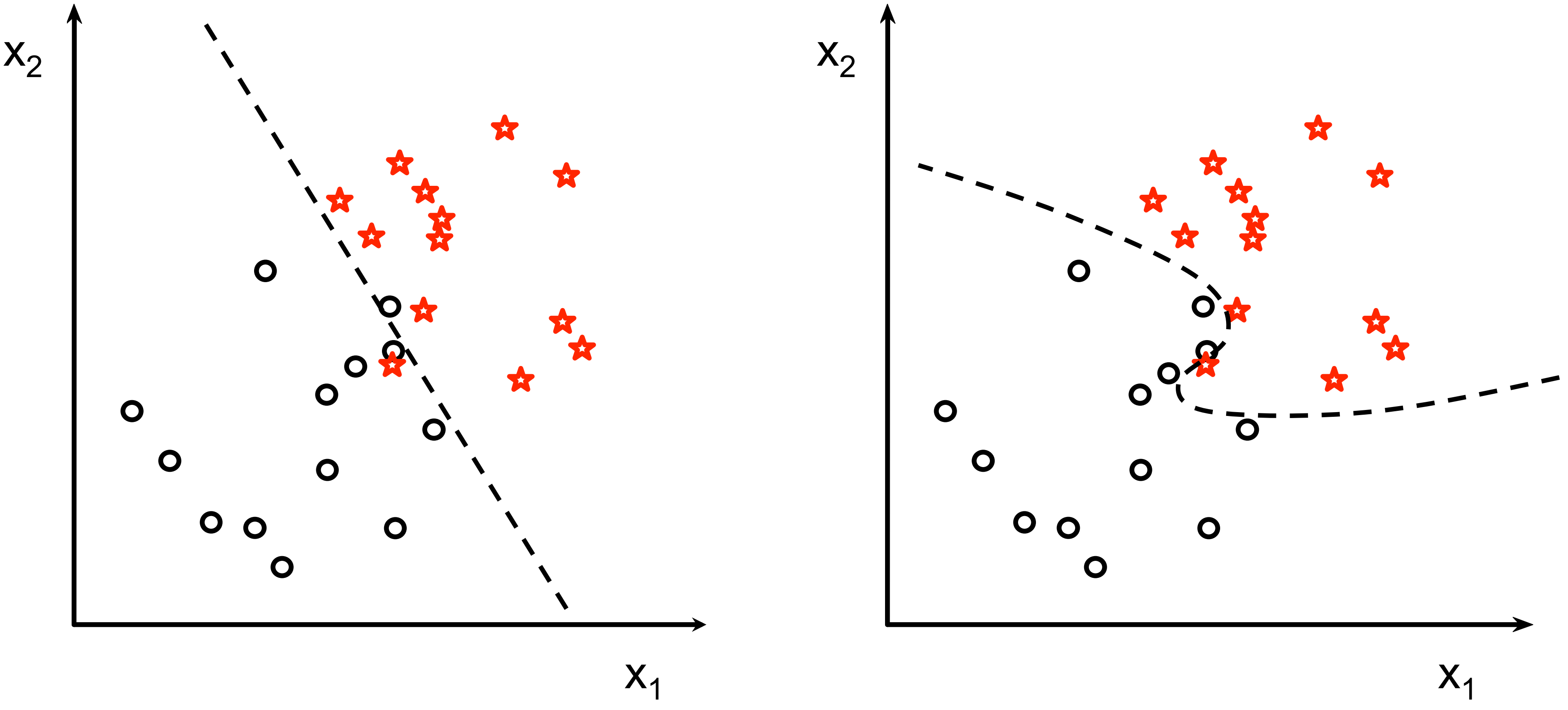
Non-linear optimum margin classifier



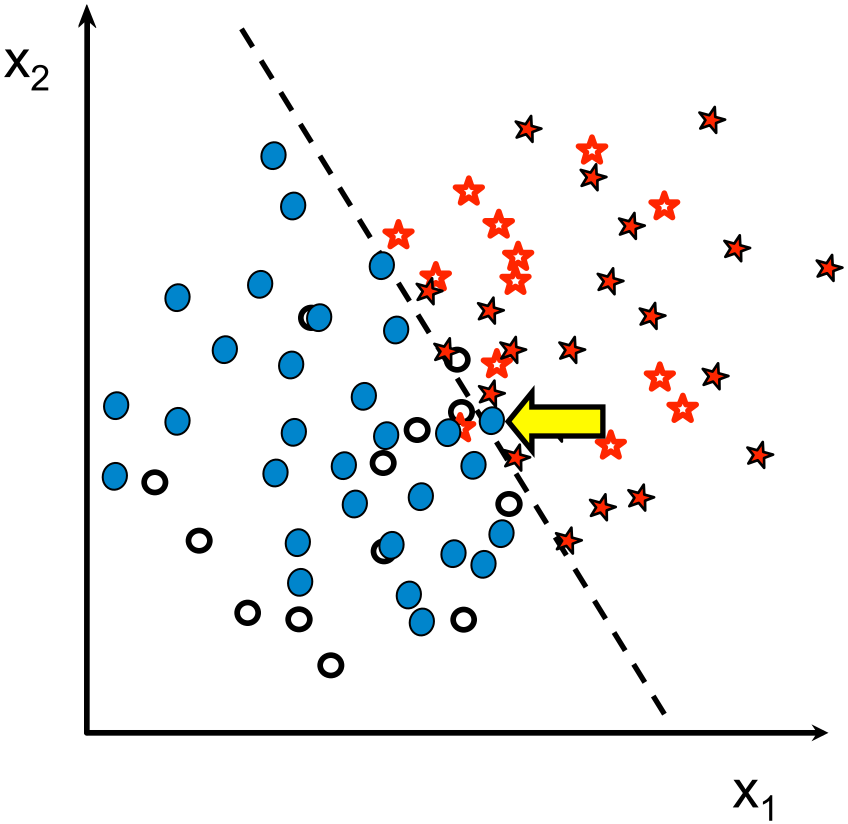
$f(x) = \sum_k \alpha_k \, k(x_k, x) + b$

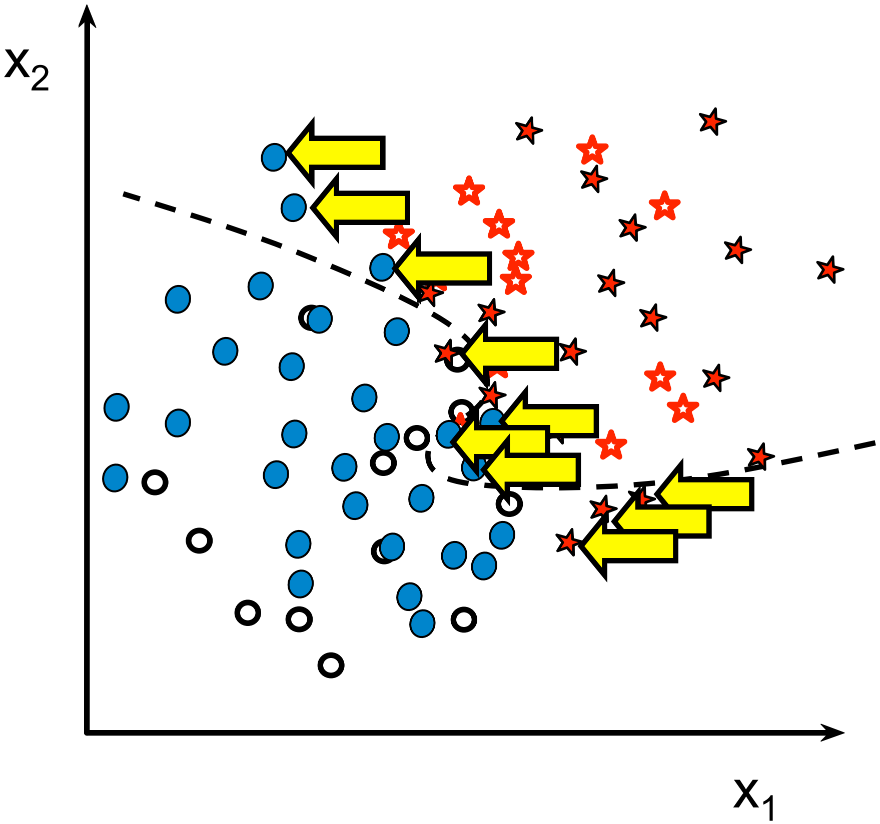
SVM, Boser-Guyon-Vapnik, 1992

Fit / Robustness Tradeoff



Fit / Robustness Tradeoff





Summary

- The **risk** is a function to evaluate LM performance.
- The risk can be optimized with training examples using **gradient descent**.
- **Stochastic gradient** mean updates are performed one example at a time.
- Perceptron update rules are the same Hebb's rule: $\Delta w = \eta\, y\, \mathbf{\Phi}(\mathbf{x})$, and $\Delta\alpha_k = \eta\, y^k$, with some conditions on the functional margin $z = y\, f(x)$:
  - $z < 0$ (regular perceptron)
  - $z < 1$ (large margin Perceptron)
  - $\min(z)$ (optimum margin Perceptron)

Come to my office hours…
Wed 2:30-4:30 Soda 329

Next time

For Ockham to Vapnik

- Shave off unnecessary parameters.
- Forget the unnecessary memories.
- Minimize complexity.
- Minimize ||**w**||.