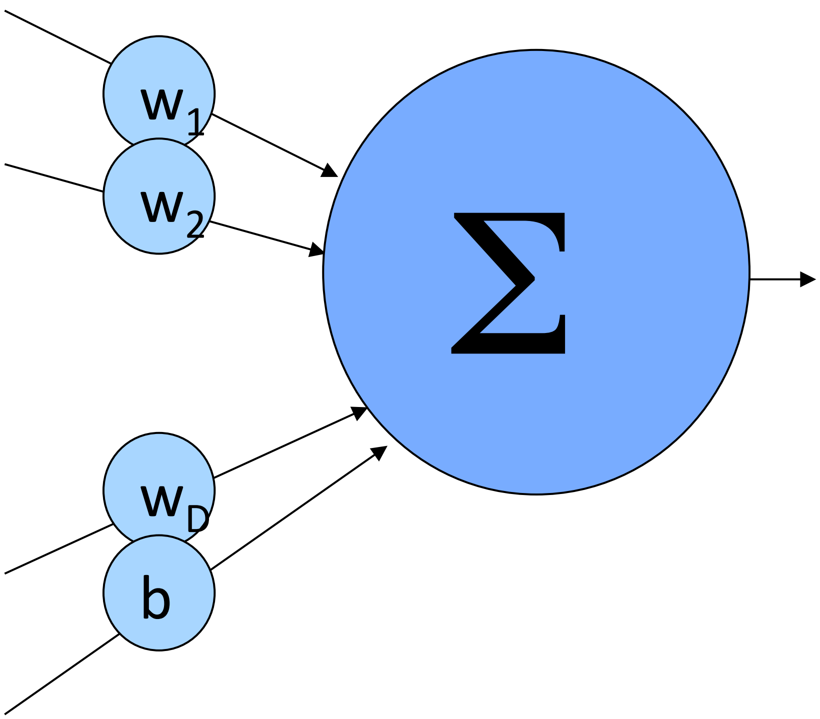
# UCB - CS189
# Introduction to Machine Learning
# Fall 2015
## Lecture 8: Kernel machines

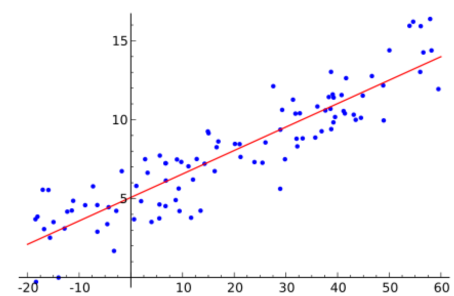
Isabelle Guyon

ChaLearn

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

## Last time




**Linear regression**

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

## Today

### Kernel machines

**PARAMETRIC (Perceptons)**

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

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

(Large margin) Perceptron

$\Delta\mathbf{w} \sim y_k \, \Phi(\mathbf{x}^k)$ if $y_k f(\mathbf{x}^k) < 1$

$\sim \mathbf{1}(1-z_k) \, y_k \, \Phi(\mathbf{x}^k)$ $\quad z_k = y_k f(\mathbf{x}^k)$

(Rosenblatt 1958)

Logistic regression

$\Delta\mathbf{w} \sim S(-z_k) \, y_k \, \Phi(\mathbf{x}^k)$

(Cox 1958)

LMS regression or classification

$\Delta\mathbf{w} \sim (y_k - f(\mathbf{x}^k)) \, \Phi(\mathbf{x}^k) \sim (1 - z_k) \, y_k \, \Phi(\mathbf{x}^k)$

(Widrow-Hoff, 1960)

**NON PARAMETRIC (Kernel machines)**

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

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

Potential Function algorithm

$\Delta\alpha_k \sim y_k$ if $y_k f(\mathbf{x}^k) < 1$

$\sim \mathbf{1}(1-z_k) \, y_k$

(Aizerman et al 1964)

Dual logistic regression

$\Delta\alpha_k \sim S(-z_k) \, y_k$

Dual LMS

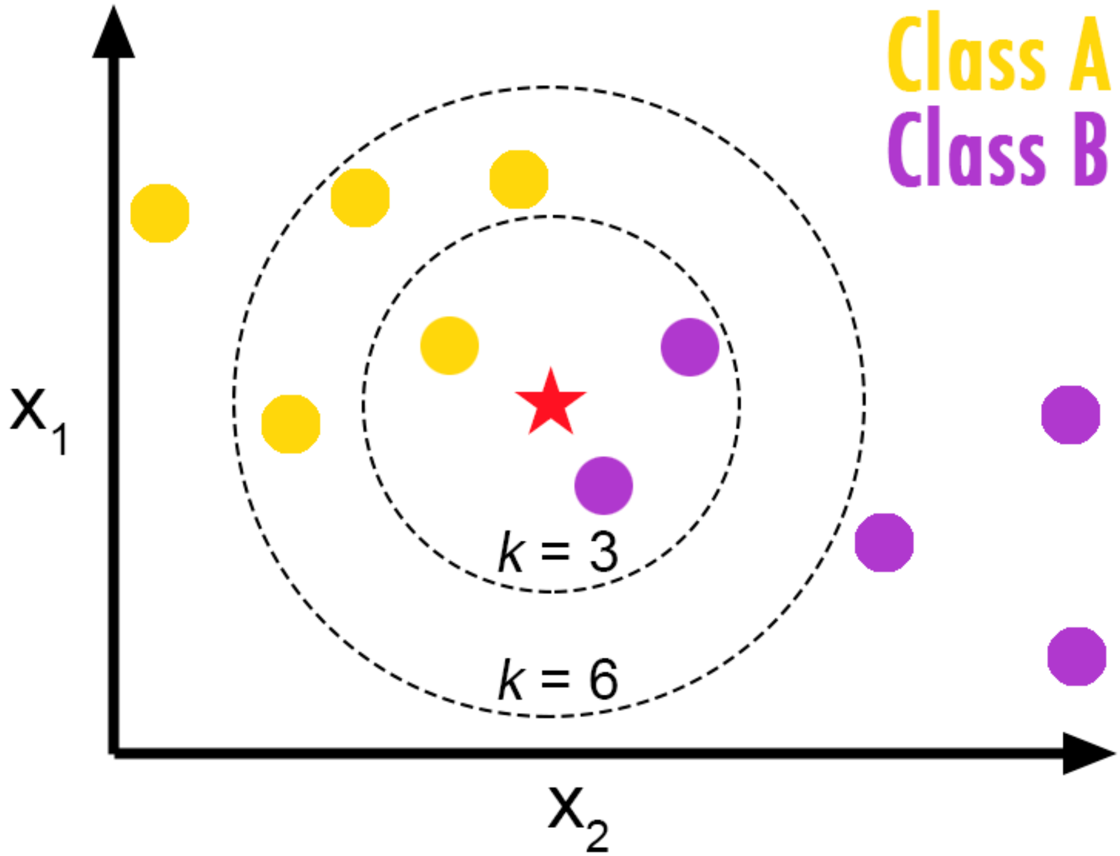
$\Delta\alpha_k \sim (y_k - f(\mathbf{x}^k)) \sim (1 - z_k) \, y_k$

4

# Linear regression

- **Problem setting:**
  - N training input/output pairs $\{(\mathbf{x}^k, y^k)\}$; y is continuous.
  - Linear model $f(\mathbf{x}) = \mathbf{w.x}$ ($w_0$ corresponds to $x_0 \equiv 1$, $\dim(\mathbf{x})=d$)
  - Solve a system of N equations $y^k = w_0 + w_1\, x_0 + \ldots$ of (d +1) unknown.
- **Matrix notation:**
  - $X\, \mathbf{w}^T = \mathbf{y}$
  - $X = [x^k{}_i]$ of dim(N,d); $\mathbf{y} = [y^k]$ of dim(N,1); $\mathbf{w} = [w_i]$ of dim(1,d).
- **Solution:**
  - Case 1: N > d, over-determined, no exact solution. $\mathbf{w}^T = X^+\mathbf{y}$ is the best solution in the least-square sense. $RSS=\Sigma_k(f(\mathbf{x})-y^k)^2$.
  - Case 2: N < d, under-determined, $\mathbf{w}^T = X^+\mathbf{y}$ is the solution of min $\|\mathbf{w}\|$.
- **Pseudo-inverse:**
  - $X^+ = \lim_{\lambda\to 0+} (X^TX + \lambda I)^{-1} X^T = \lim_{\lambda\to 0+} X^T (X\, X^T + \lambda I)^{-1}$
  - Inverse the smallest matrix and adjust $\lambda>0$ by cross-validation (see SVD trick).
- **Kernel trick:**
  - in case 2, N<<d, replace $X\, X^T$ by a (N, N) kernel matrix $K = k(\mathbf{x}^k, \mathbf{x}^h)$
  - $\boldsymbol{\alpha} = (K+ \lambda I)^{-1}\, \mathbf{y}$, gives you a non linear regression function $f(\mathbf{x}) = \Sigma_k\, \alpha_k\, k(\mathbf{x}, \mathbf{x}_k)$.

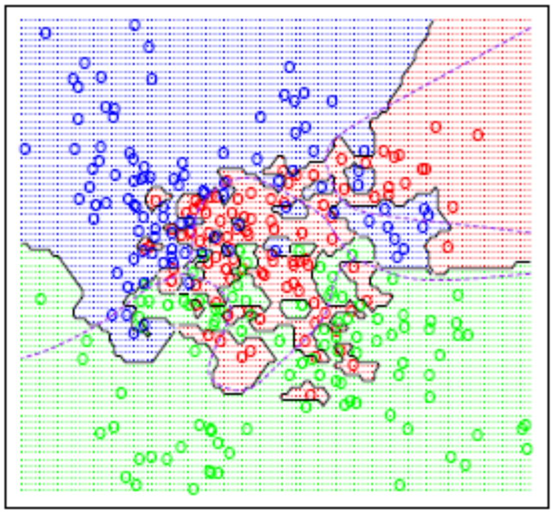
# Ancestor of kernel methods: KNN



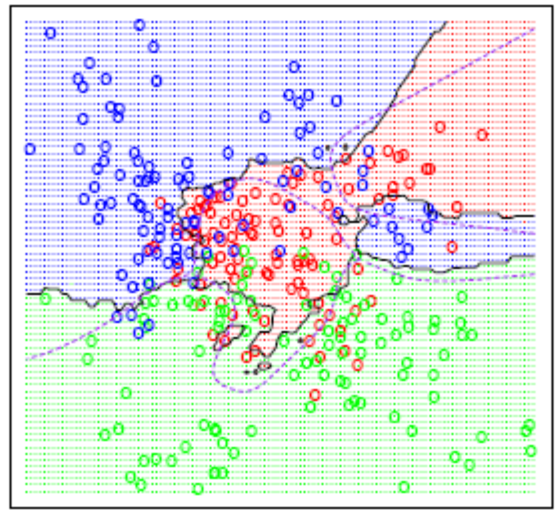

Assign to ★ the class label of the majority of the k closest examples.

Source: http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/

# Optimum k in k-nearest neighbors



k=1



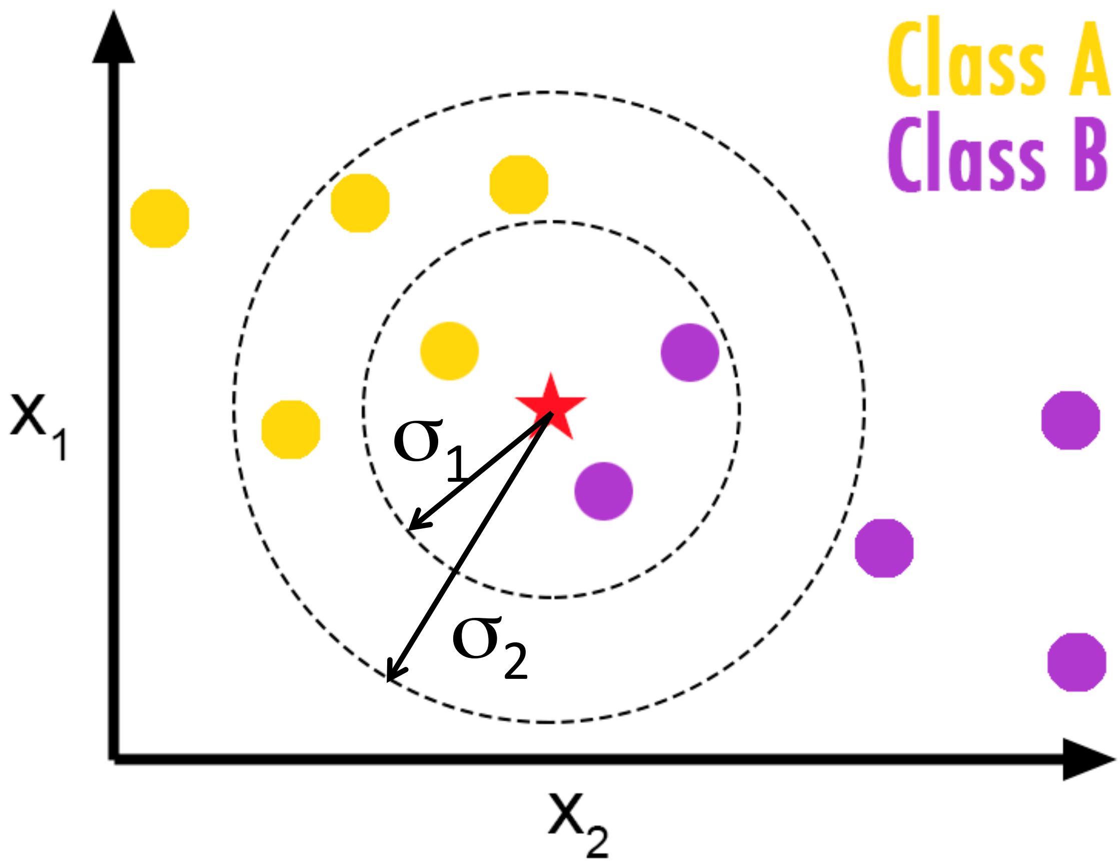
k=15

**Fit vs. robustness tradeoff in kNN:**

- kNN does not have any parameters (non parametric method).
- The "complexity" (of the decision function) *decreases* with k.
- k is a *hyper-parameter* adjustable by *cross-validation.*
- kNN can also be used for regression.

Source: https://onlinecourses.science.psu.edu/stat857/node/21

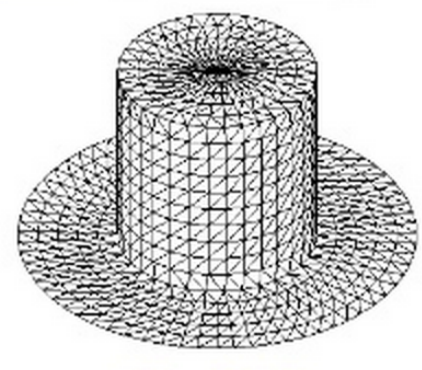
# Parzen windows




Assign to ★ the class label of the majority of the examples enclosed in a sphere of radius $\sigma_r$.

# Parzen windows is a kernel method

- Assign to **x** the class label of the majority of the examples enclosed in a sphere of radius $\sigma$.
- $f(\mathbf{x}) = \sum_{k=1:N} y_k \, k(\mathbf{x}, \mathbf{x}_k)$

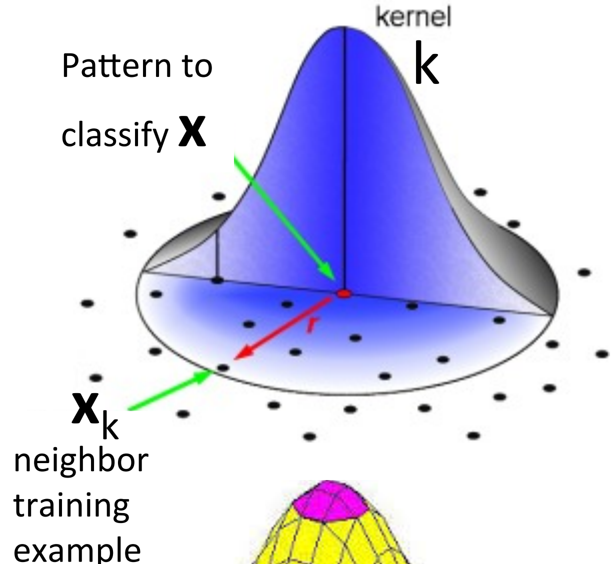


- Top hat kernel:

$k(\mathbf{x}, \mathbf{x}_k) = \mathbf{1}(\|\mathbf{x}-\mathbf{x}_k\|^2 < \sigma^2)$

# Parzen windows is a kernel method

- Assign to **x** the class label of the majority of the examples enclosed in a sphere of radius $\sigma$.
- $f(\mathbf{x}) = \sum_{k=1:N} y_k \, k(\mathbf{x}, \mathbf{x}_k)$

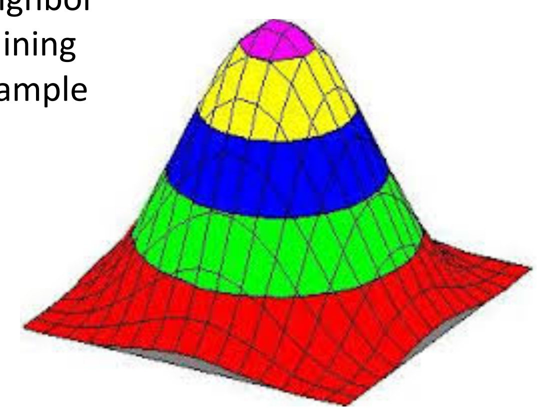



- <u>Top hat kernel:</u>

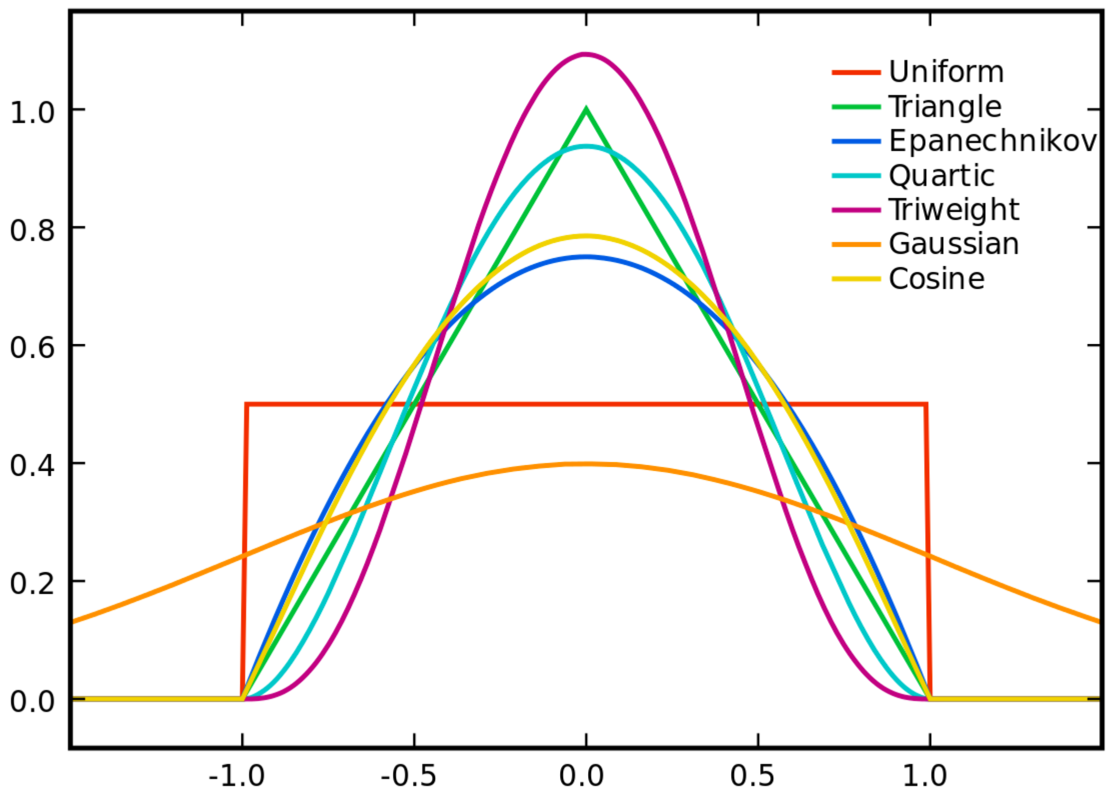
$k(\mathbf{x}, \mathbf{x}_k) = \mathbf{1}(\|\mathbf{x}-\mathbf{x}_k\|^2 < \sigma^2)$

- <u>Gaussian kernel:</u>

$k(\mathbf{x}, \mathbf{x}_k) = \exp -(\|\mathbf{x}-\mathbf{x}_k\|^2 / 2\sigma^2)$



# Some radial kernels




Source: https://en.wikipedia.org/wiki/Kernel_(statistics)
See also: http://crsouza.com/2010/03/kernel-functions-for-machine-learning-applications
http://www.cs.berkeley.edu/~bartlett/courses/281b-sp08/8.pdf

# Non radial kernels

$$f(\mathbf{x}) = \sum_{k=1:N} y_k \, k(\mathbf{x}, \mathbf{x}_k)$$

- Linear kernel:

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

The Parzen windows $f(\mathbf{x})$ for the linear kernel is just Hebb's rule!

- Polynomial kernel:

$k(\mathbf{x}, \mathbf{x}_k) = (1+ \mathbf{x}.\mathbf{x}_k )^q$

# What makes k(**x**, **x**') a "good" kernel?

- Symmetric k(**x**, **x'**) = k(**x'**, **x**)
- Kernel matrix K = [k($\mathbf{x}_k$, $\mathbf{x}_h$)] $_{k=1:N,\ h=1:N}$ invertible, possibly after "regularization" (K+$\lambda$I), $\lambda$>0. Satisfied if all eigenvalues≥0 (PSD matrix).
- True in particular if:
  – k(**x**, **x'**) is a PSD kernel i.e. satisfies Mercer's condition.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_i, x_j) c_i c_j \geq 0$$

for all finite sequences of points $x_1, \ldots, x_n$ of $[a, b]$ and all choices of real numbers $c_1, \ldots, c_n$

  There exists an eigen decomposition k(**x**, **x'**) = $\sum_{i=1:\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}')$
  – K is a Gram matrix (outer product K = $XX^T$ or K = $\Phi\Phi^T$)
  – K is a covariance matrix.
- The sigmoid kernel tanh(**x**.**x'**) is NOT PSD.

# Remember the "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**

# Reproducing kernels

- A Hilbert space H is a space of functions $\{f(\mathbf{x})\}$.
- A "reproducing kernel" Hilbert space (RKHS) is endowed with a dot product:

$$<f, g>_H = \int f(\mathbf{x})\, g(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}$$

- H has a (unique) positive definite "reproducing kernel":

$$k(\mathbf{s}, \mathbf{t}) = <k(., \mathbf{s}), k(., \mathbf{t})>_H$$

- Reproducing kernel property:

$$f(\mathbf{x}) = <k(., \mathbf{x}), f>_H$$

See for details: http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/RKHS_Notes1.pdf

# Representer theorem

Kimeldorf, George S.; Wahba, Grace (1970)
Schölkopf, Bernhard; Herbrich, Ralf; Smola, Alex J. (2001)

- $f(\mathbf{x})$ is a function in RKHS H, w. $<f, g>_H = \int f(\mathbf{x})\, g(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}$, and kernel k: $f(\mathbf{x}) = <k(., \mathbf{x}), f>_H$ .
- Given a risk functional $R[f]$ and a regularizer $\Omega[f]$ and training examples $\{\mathbf{x}^k, \mathbf{y}^k\}$ on which we evaluate $R[f] + \lambda\Omega[f]$, $\lambda>0$.
- $f^*(\mathbf{x}) = \text{argmin}_f\, R_{train}[f] + \lambda\Omega[f]$ admits a representation

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

- This means that we only have to solve for $\alpha_k$ and that the solution lies in the span of $k(., \mathbf{x}^k)$ that can be thought of as special kind of features. The kernel determines the type of function used and the type of regularization (smoothness).

See for details: https://en.wikipedia.org/wiki/Representer_theorem

# Mercer kernels and kernel trick

$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1:\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}')$

$\langle f, g \rangle = \int f(\mathbf{s})\, g(\mathbf{s})\, p(\mathbf{s})\, d\mathbf{s}$ ; $\psi_i(\mathbf{x})$ are eigen functions, $\langle \psi_i\, \psi_j \rangle = \delta_{ij}$

$\lambda_i \psi_i(\mathbf{x}) = \langle k(.,\mathbf{x}), \psi_i \rangle$ with eigen values $\lambda_i > 0$

$\langle f, g \rangle_H = \sum_{i=1:\infty} (1/\lambda_i) \langle f, \psi_i \rangle \langle g, \psi_i \rangle$

Use $\phi_i(\mathbf{x}) = \sqrt{\lambda_i}\, \psi_i(\mathbf{x})$

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

$$= \sum_i \left( \sum_k \alpha_k \phi_i(\mathbf{x}^k) \right) \phi_i(\mathbf{x})$$

$$= \sum_i w_i \phi_i(\mathbf{x})$$

See for details: https://en.wikipedia.org/wiki/Representer_theorem

# Kernel algebra

Cristianini and Shaw-Taylor, 2001

**Proposition 3.22 (Closure properties)** *Let $\kappa_1$ and $\kappa_2$ be kernels over $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(\cdot)$ a real-valued function on $X$, $\phi\colon X \longrightarrow \mathbb{R}^N$ with $\kappa_3$ a kernel over $\mathbb{R}^N \times \mathbb{R}^N$, and $\mathbf{B}$ a symmetric positive semi-definite $n \times n$ matrix. Then the following functions are kernels:*

(i) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$,
(ii) $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$,
(iii) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$,
(iv) $\kappa(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$,
(v) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_3(\phi(\mathbf{x}),\phi(\mathbf{z}))$,
(vi) $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}'\mathbf{B}\mathbf{z}$.

**Proposition 3.24** *Let $\kappa_1(\mathbf{x}, \mathbf{z})$ be a kernel over $X \times X$, where $\mathbf{x}, \mathbf{z} \in X$, and $p(x)$ is a polynomial with positive coefficients. Then the following functions are also kernels:*

(i) $\kappa(\mathbf{x}, \mathbf{z}) = p(\kappa_1(\mathbf{x}, \mathbf{z}))$,
(ii) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(\kappa_1(\mathbf{x}, \mathbf{z}))$,
(iii) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/(2\sigma^2))$.

http://cs.du.edu/~mitchell/mario_books/Kernel_Methods_for_Pattern_Analysis_-_John_Shawe-Taylor_&_Nello_Christianini.pdf

# Favorite kernel

$k(\mathbf{x}, \mathbf{x}') = (a + \mathbf{x} \cdot \mathbf{x}')^b \exp(- d \, \|\mathbf{x}-\mathbf{x}_k\|^c)$

Of 4 hyper-parameters a, b, c, d.

Simplify: take a = 1 and c = 2.
Then: b = 0 → Gaussian kernel
d = 0 → Polynomial kernel

Adjust the hyper-parameters by cross-validation.

# Kernel machines

**PARAMETRIC (Perceptons)**

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

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

(Large margin) Perceptron

$\Delta\mathbf{w} \sim y_k \Phi(\mathbf{x}^k)$ if $y_k f(\mathbf{x}^k)<1$

$\sim \mathbf{1}(1\text{-}z_k)\, y_k \Phi(\mathbf{x}^k)$ $z_k = y_k f(\mathbf{x}^k)$

(Rosenblatt 1958)

Logistic regression

$\Delta\mathbf{w} \sim S(-z_k)\, y_k\, \mathbf{\Phi}(\mathbf{x}^k)$

(Cox 1958)

LMS regression or classification

$\Delta\mathbf{w} \sim (y_k - f(\mathbf{x}^k))\, \mathbf{\Phi}(\mathbf{x}^k) \sim (1- z_k)\, y_k\, \mathbf{\Phi}(\mathbf{x}^k)$

(Widrow-Hoff, 1960)

**NON PARAMETRIC (Kernel machines)**

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

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

Potential Function algorithm

$\Delta\alpha_k \sim y_k$ if $y_k f(\mathbf{x}^k)<1$

$\sim \mathbf{1}(1\text{-}z_k)\, y_k$

(Aizerman et al 1964)

Dual logistic regression

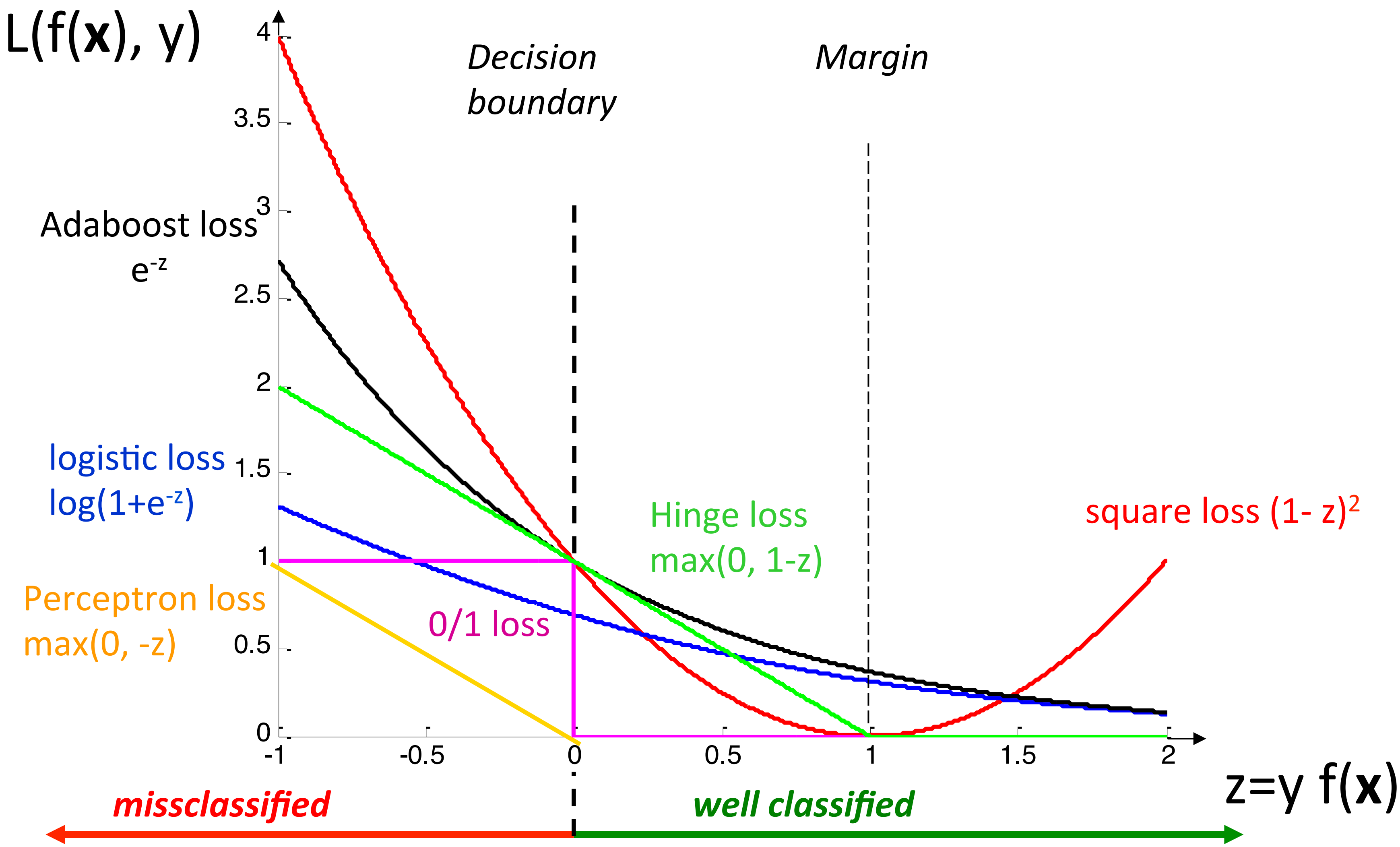
$\Delta\alpha_k \sim S(-z_k)\, y_k$

Dual LMS

$\Delta\alpha_k \sim (y_k - f(\mathbf{x}^k)) \sim (1- z_k)\, y_k$

# Loss Functions

The risk is the average of the loss.



# Regularized risk minimization

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

- <u>Minimize:</u>

  $R_{reg}[\mathbf{w}] = \sum_k L(f(\mathbf{x}^k), y_k) + \lambda \, \Omega(\mathbf{w})$

  $\Omega(\mathbf{w}) = \|\mathbf{w}\|^2$

  $\Omega(\mathbf{w}) = \|\mathbf{w}\|$

# Stochastic gradient

$f(\mathbf{x}) = \Sigma_j w_j \Phi_j(\mathbf{x}) = \mathbf{w} \bullet \mathbf{\Phi}(\mathbf{x}) = \Sigma_k \alpha_k k(\mathbf{x}^k, \mathbf{x})$

1) Compute $\partial L/\partial w_i = \partial L/\partial f \,.\, \partial f/\partial w_i = \partial L/\partial f \; \Phi_i(\mathbf{x})$

- Classification: $\partial L/\partial w_i = \partial L/\partial z . \partial z/\partial w_i$ with $z = y f(\mathbf{x})$. $\partial z/\partial w_i = y \Phi_i(\mathbf{x}) \rightarrow \partial L/\partial w_i = \partial L/\partial z \,.\, y \Phi_i(\mathbf{x})$

2) Compute $\partial\Omega/\partial w_i$

- $\Omega(\mathbf{w}) = \|\mathbf{w}\|^2 \quad \rightarrow \qquad \partial\Omega/\partial w_i = 2w_i$

3) Compute the negative gradient of $L(f(\mathbf{x}), y)+\lambda\Omega(\mathbf{w})$

$\Delta w_i = -\eta \, \partial L/\partial f \; \Phi_i(\mathbf{x}) - \gamma \, w_i$

$$\boxed{\Delta \mathbf{w} = - \eta \, \partial L/\partial f \; \mathbf{\Phi}(\mathbf{x}) - \gamma \, \mathbf{w}}$$

# Kernel version

$\Phi$−space version: $\Delta \mathbf{w} = - \eta\ \partial L/\partial f\ \Phi(\mathbf{x})$

- Learning $(\mathbf{x}^k, y^k)$: $\Delta \mathbf{w} = - \eta\ \partial L/\partial f |_{(\mathbf{x}^k, y^k)}\ \Phi(\mathbf{x}^k)$
- $\mathbf{w} = \Sigma_k\ \alpha_k\ \Phi(\mathbf{x}^k)$, $\Delta \mathbf{w} = \Delta \alpha_k\ \Phi(\mathbf{x}^k)$

$$\boxed{\Delta \alpha_k = - \eta\ \partial L/\partial f |_{(\mathbf{x}^k, y^k)}}$$

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

# Kernel version with shrinkage

$\Phi$–space version: $\Delta \mathbf{w} = - \eta\ \partial L/\partial f\ \Phi(\mathbf{x}) - \gamma\ \mathbf{w}$

- Learning $(\mathbf{x}^k, y^k)$:

$$\Delta \mathbf{w} = - \eta\ \partial L/\partial f|_{(\mathbf{x}^k, y^k)}\ \Phi(\mathbf{x}^k) - \gamma\ \mathbf{w}$$

- $\mathbf{w} = \Sigma_k\ \alpha_k\ \Phi(\mathbf{x}^k)$, $\Delta \mathbf{w} = \Delta\alpha_k\ \Phi(\mathbf{x}^k)$

| | |
|---|---|
| $\Delta\alpha_k = - \eta\ \partial L/\partial f|_{(\mathbf{x}^k, y^k)} - \gamma\ \alpha_k$ | for example k |
| $\Delta\alpha_h = -\gamma\ \alpha_h$ | for other examples |

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

# Example: Least Mean Square or LMS

*Widrow-Hoff, 1960*

- $L_{square\_loss} = (f(\mathbf{x}) - y)^2$
- $L/\partial f = 2\ (f(\mathbf{x}) - y)$
- $\partial f/\partial w_{i\ =}\ \Phi_i(\mathbf{x})$ for $f(\mathbf{x}) = \Sigma_j\ w_j\ \Phi_j(\mathbf{x})$
- $\partial L/\partial w_i = 2\ (f(\mathbf{x}) - y)\ \Phi_i(\mathbf{x})$ Loss variation
- $\Delta w_i = -\ \eta\ (f(\mathbf{x}) - y)\ \Phi_i(\mathbf{x}) - \gamma\ \mathbf{w}$

$$\boxed{\Delta \mathbf{w} = \eta\ (y - f(\mathbf{x}))\ \Phi(\mathbf{x}) - \gamma\ \mathbf{w}}$$ LMS learning rule

$\Delta \alpha_k = \eta\ (y^k - f(\mathbf{x}^k))\ -\gamma\ \alpha_k$ for example k

$\Delta \alpha_h = -\gamma\ \alpha_h$ for other examples

# Comparison Hebbs's rule

- $f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x})$
- <u>Hebb's rule:</u>

  $\Delta w_i = \eta \, y \, \Phi_i(\mathbf{x})$

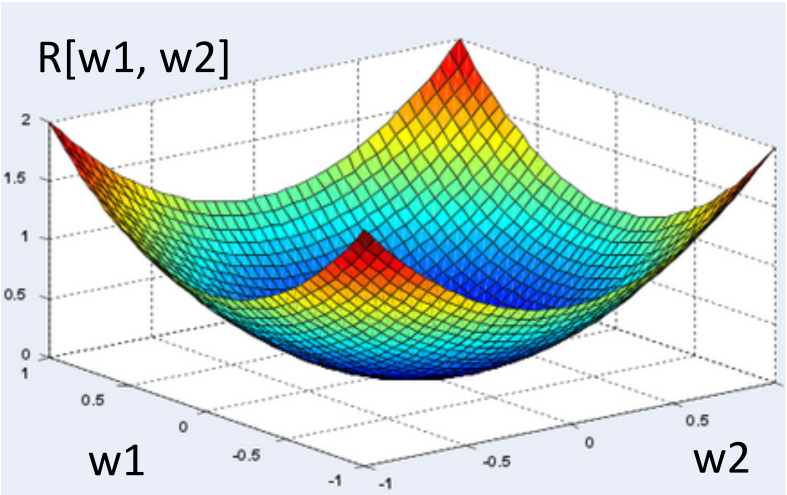
  $\Delta \alpha_k = \eta \, y^k$
- <u>LMS rule:</u> 1

  $\Delta w_i = \eta \, (y - f(\mathbf{x})) \, \Phi_i(\mathbf{x})$

  $\Delta \alpha_k = \eta \, (y^k - f(\mathbf{x}^k))$

# Exploiting convexity?

- Stochastic gradient does not exploit convexity, is this the best approach?
  - Yes for BIG data.
  - No if either N or d is small.




- For regression, regularized inverse:

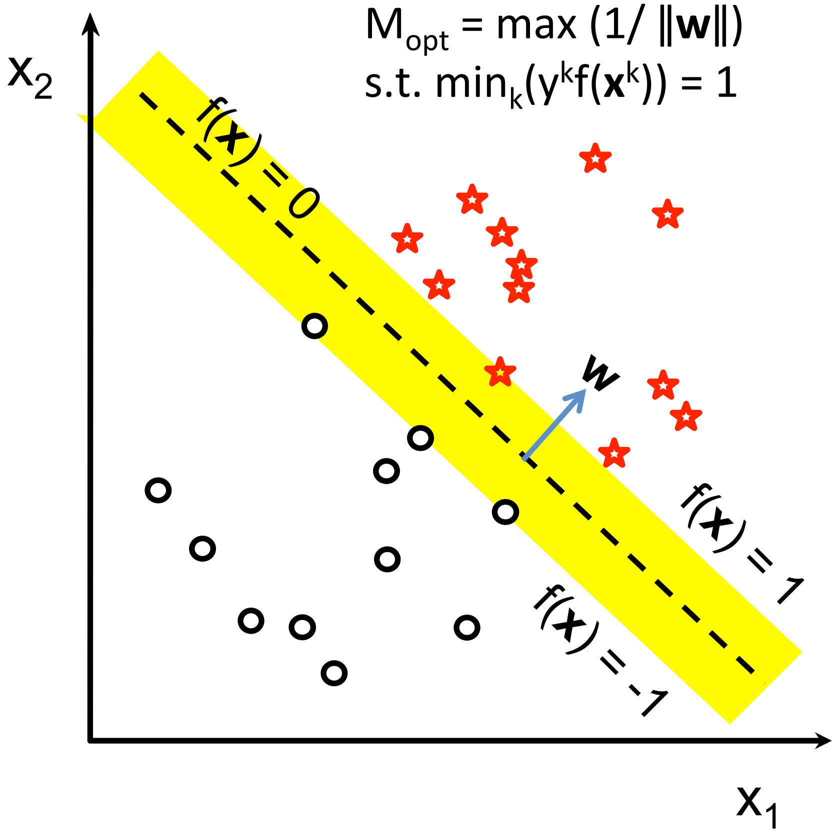
$$\mathbf{w}^T = (X^TX + \lambda I)^{-1} X^T \mathbf{y} \text{ or } X^T (X X^T + \lambda I)^{-1} \mathbf{y}$$

$$\alpha = (K+\lambda I)^{-1} \mathbf{y}$$

# Classification with optimum margin

## Hard margin

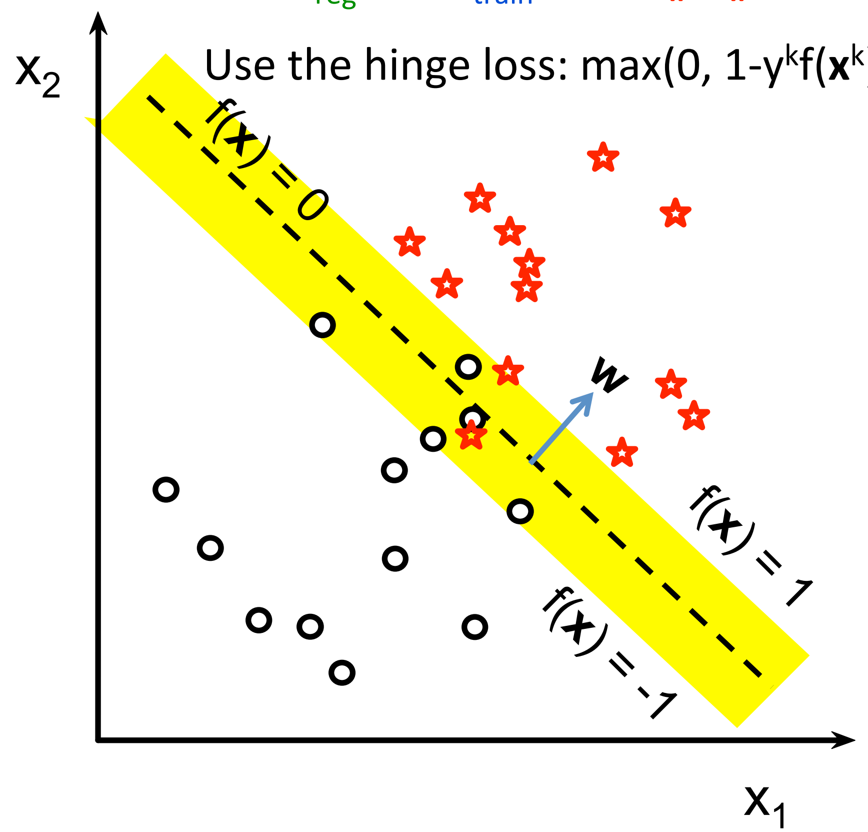
$M = 1/\ \|\mathbf{w}\|$
$M_{opt} = \max\ (1/\ \|\mathbf{w}\|)$
s.t. $\min_k(y^k f(\mathbf{x}^k)) = 1$




## Soft margin

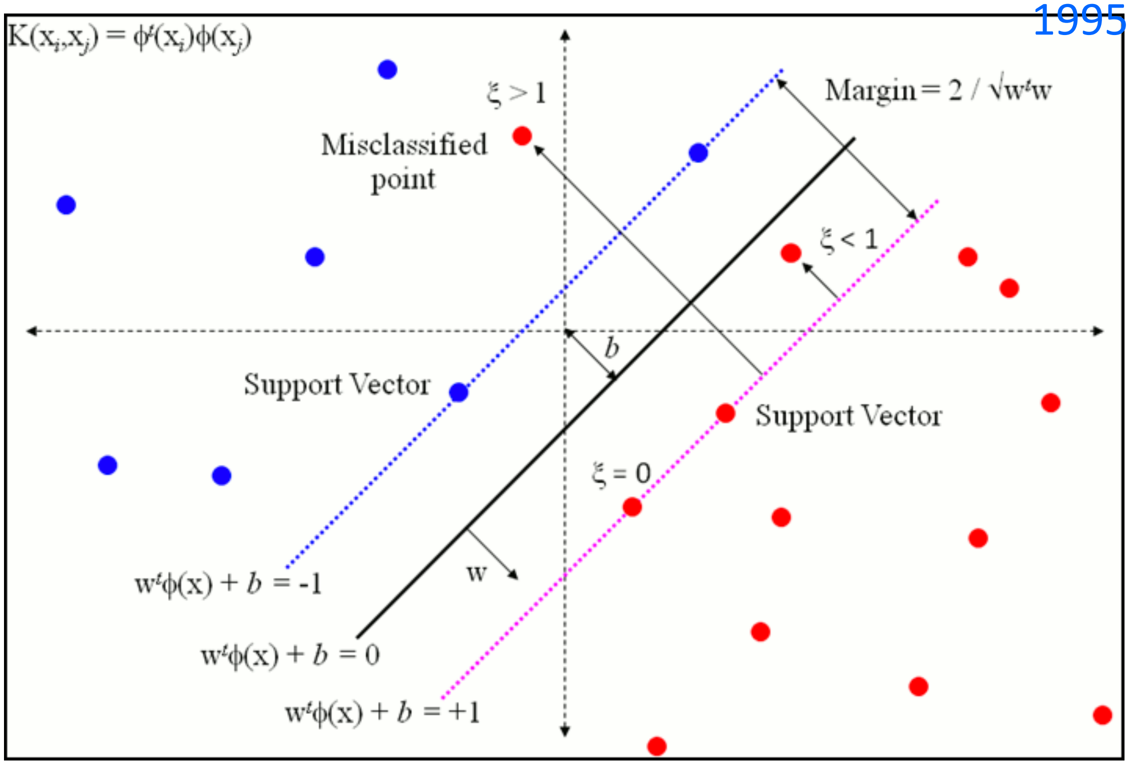
$\min R_{reg}[f] = R_{train}[f] + \lambda\ \|\mathbf{w}\|^2$

Use the hinge loss: $\max(0, 1\text{-}y^k f(\mathbf{x}^k))$



# Soft margin

Cortes-Vapnik, 1995




min $R_{reg}[f] = R_{train}[f] + \lambda \|\mathbf{w}\|^2$

Use the hinge loss: $\max(0, 1-y^k f(\mathbf{x}^k))$

min $C \sum_{k=1:N} \xi_k + (1/2) \|\mathbf{w}\|^2$

$1-y^k f(\mathbf{x}^k) \leq \xi_k \quad \xi_k > 0$

http://stats.stackexchange.com/questions/108617/simple-svm-question

# Standard quadratic programs

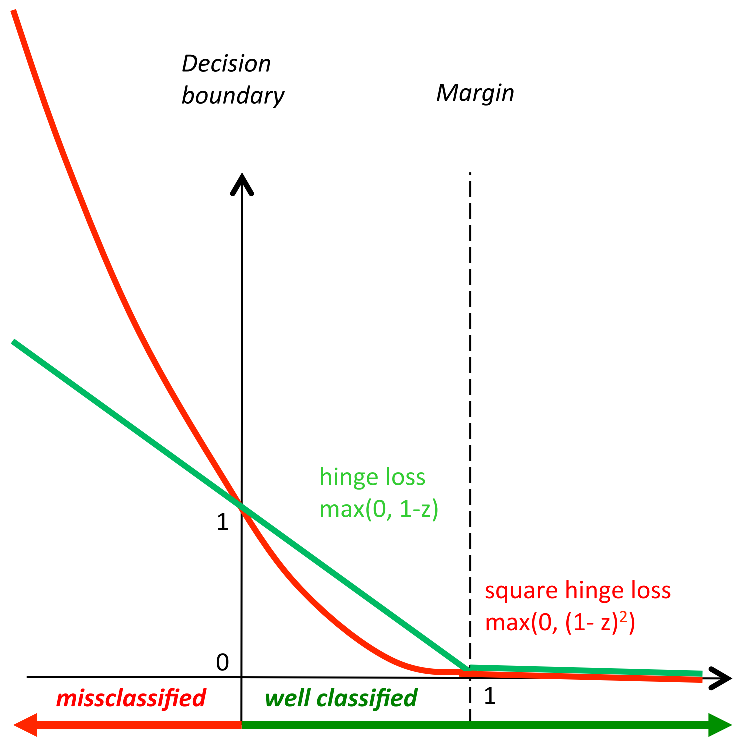
$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) + b$

1) Soft margin Perceptron-style
   (with hinge loss)

$$\begin{cases} \min_{w,b,\xi} C \sum_{k=1:N} \xi_k + (1/2) \|\mathbf{w}\|^2 \\ y^k f(\mathbf{x}^k) - 1 + \xi_k \geq 0 \\ \xi_k \geq 0 \end{cases}$$

2) Soft margin RSS-style
   (with square hinge loss)

$$\begin{cases} \min_{w,b,\xi} C \sum_{k=1:N} (\xi_k)^2 + (1/2) \|\mathbf{w}\|^2 \\ y^k f(\mathbf{x}^k) - 1 + \xi_k \geq 0 \\ \xi_k \geq 0 \end{cases}$$



# Dual problems (in $\alpha$ space)

- $f(\mathbf{x}) = \sum_k \alpha_k\, y_k\, k(\mathbf{x}^k, \mathbf{x})$ $\quad$ $H = [y_k y_h\, k(\mathbf{x}^k, \mathbf{x}^h)]$

1) Hinge loss version:

$$\min_\alpha \boldsymbol{\alpha}^T \mathbf{1} - (1/2)\, \boldsymbol{\alpha}^T H \boldsymbol{\alpha}$$

$0 \le \alpha_k \le C$ box constraint

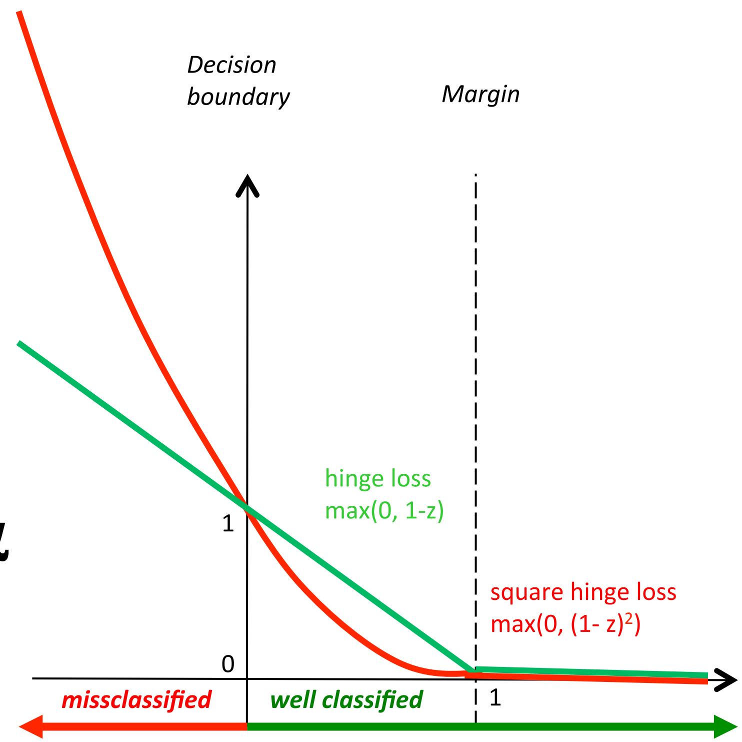
$\boldsymbol{\alpha}^T \mathbf{y} = 0$ bias constraint

2) Square hinge loss version

$$\min_\alpha \boldsymbol{\alpha}^T \mathbf{1} - (1/2)\, \boldsymbol{\alpha}^T (H + (1/C)\, \mathrm{I}) \boldsymbol{\alpha}$$

$\alpha_k \ge 0$

$\boldsymbol{\alpha}^T \mathbf{y} = 0$ bias constraint




http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf
http://cs229.stanford.edu/notes/cs229-notes3.pdf

# Dual problems (in $\alpha$ space)

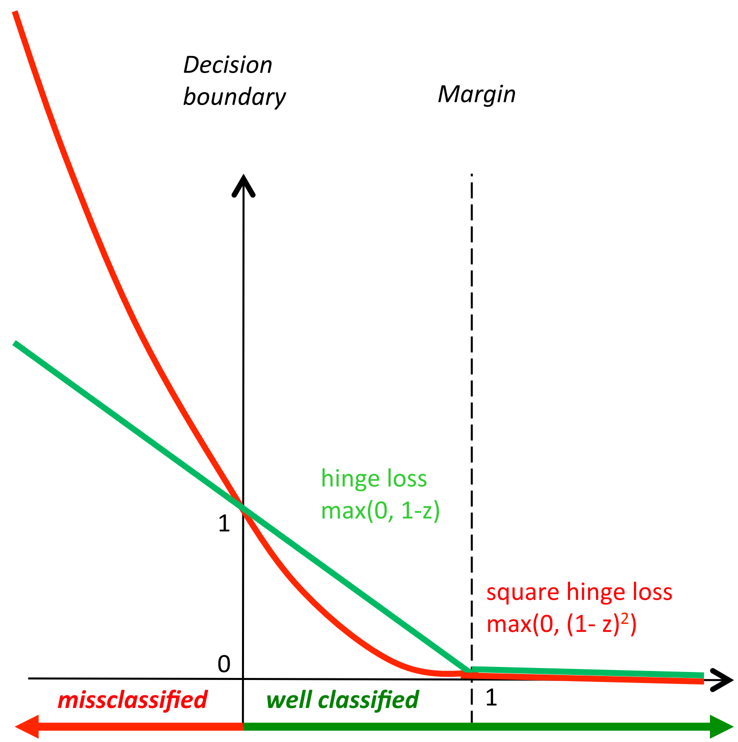
- $f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x})$ $\qquad K = [k(\mathbf{x}^k, \mathbf{x}^h)]$

1) Hinge loss version:

$$\left\{ \begin{array}{ll} \min_\alpha \boldsymbol{\alpha}^T \mathbf{y} - (1/2)\, \boldsymbol{\alpha}^T K \boldsymbol{\alpha} & \\ 0 \le \alpha_k y_k \le 1/\lambda & \text{box constraint} \\ \boldsymbol{\alpha}^T \mathbf{1} = 0 & \text{bias constraint} \end{array} \right.$$

2) Square hinge loss version

$$\left\{ \begin{array}{ll} \min_\alpha \boldsymbol{\alpha}^T \mathbf{y} - (1/2)\, \boldsymbol{\alpha}^T (K + \lambda I) \boldsymbol{\alpha} & \\ \alpha_k y_k \ge 0 & \text{ridge} \\ \boldsymbol{\alpha}^T \mathbf{1} = 0 & \text{bias constraint} \end{array} \right.$$




http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf
http://cs229.stanford.edu/notes/cs229-notes3.pdf

# Summary

- $f(\mathbf{x}) = \sum_{k=1:N} \alpha_k \, k(\mathbf{x}, \mathbf{x}_k)$
- Kernel methods are inspired by non-parameteric example-based methods; each example $\mathbf{x}_k$ votes according to:
  - How similar it is to $\mathbf{x}$ measured by $k(\mathbf{x}, \mathbf{x}_k)$
  - It weight $\alpha_k$
- Kernels are "dot products" in a (possibly infinite) $\Phi$ space.
- The validity of a kernel can be checked with Mercer's condition; there are also kernel composition theorems.
- The weights $\alpha_k$ can be optimized by gradient descent (for big data) or convex optimization methods.

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

## Next time

**Performance evaluation**