Lecture Slides for

INTRODUCTION TO

Machine Learning

2nd Edition

ETHEM ALPAYDIN

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml2e

CHAPTER 13:

Kernel Machines

Kernel Machines

- Discriminant-based: No need to estimate densities first
- Define the discriminant in terms of support vectors
- The use of kernel functions, application-specific measures of similarity
- No need to represent instances as vectors
- Convex optimization problems with a unique solution

Optimal Separating Hyperplane

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_t \text{ where } r^t = \begin{cases} +1 & \text{if } \mathbf{x}^t \in C_1 \\ -1 & \text{if } \mathbf{x}^t \in C_2 \end{cases}$$

find $\mathbf{w}$ and w_0 such that

$$\mathbf{w}^T \mathbf{x}^t + w_0 \geq +1 \text{ for } r^t = +1$$

$$\mathbf{w}^T \mathbf{x}^t + w_0 \leq +1 \text{ for } r^t = -1$$

which can be rewritten as

$$r^t \left(\mathbf{w}^T \mathbf{x}^t + w_0 \right) \geq +1$$

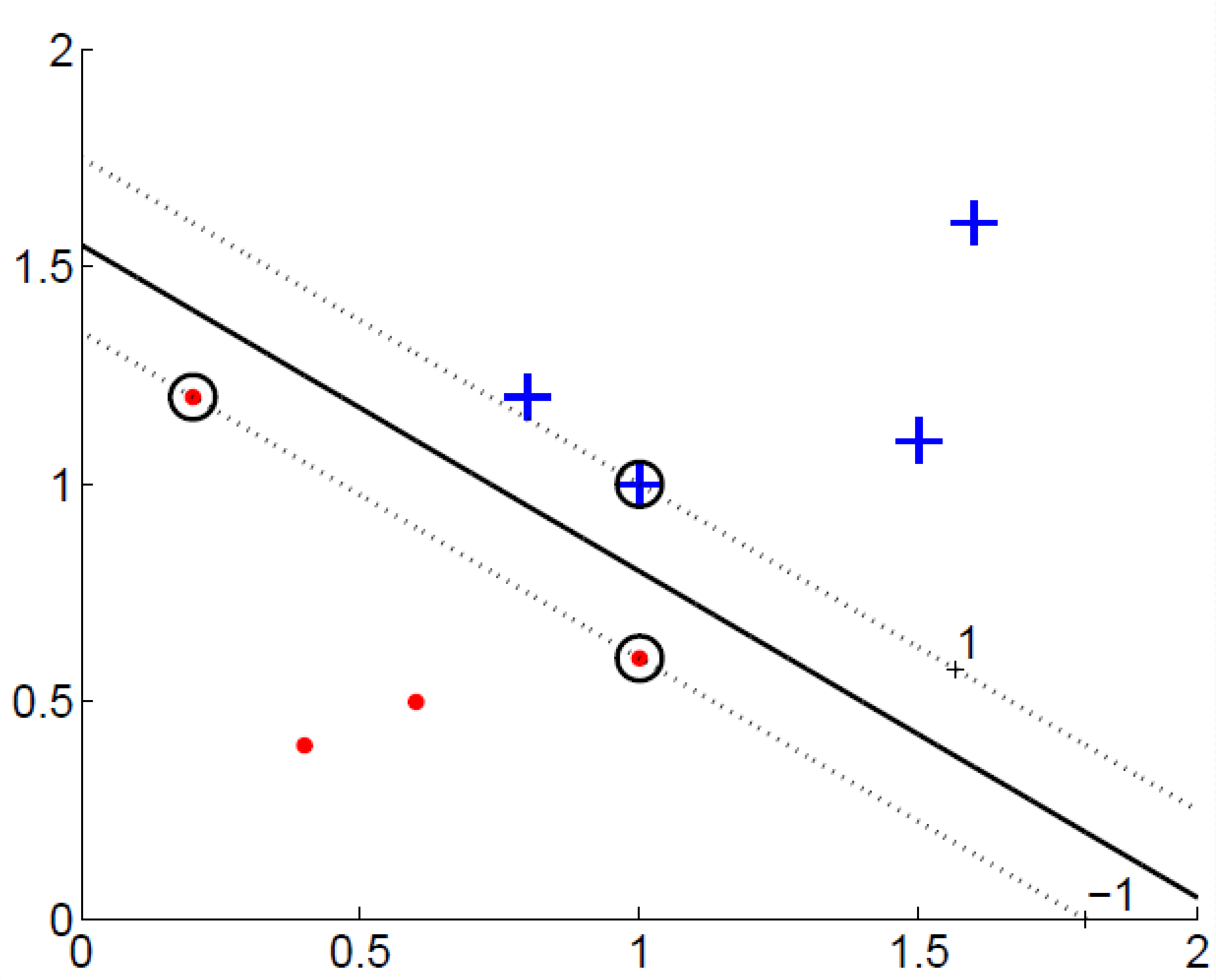
(Cortes and Vapnik, 1995; Vapnik, 1995)

Margin

- Distance from the discriminant to the closest instances on either side
- Distance of x to the hyperplane is $\dfrac{\left|\mathbf{w}^T\mathbf{x}^t + w_0\right|}{\|\mathbf{w}\|}$
- We require $\dfrac{r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right)}{\|\mathbf{w}\|} \geq \rho, \forall t$
- For a unique sol'n, fix $\rho\|\mathbf{w}\|=1$, and to max margin

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1, \forall t$$

Margin



$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \ge +1, \forall t$$

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t \left[r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) - 1\right]$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) + \sum_{t=1}^{N} \alpha^t$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{t=1}^{N} \alpha^t r^t \mathbf{x}^t$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{t=1}^{N} \alpha^t r^t = 0$$

$$L_d = \frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right) - \mathbf{w}^T\sum_t \alpha^t r^t \mathbf{x}^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t$$

$$= -\frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right) + \sum_t \alpha^t$$

$$= -\frac{1}{2}\sum_t\sum_s \alpha^t \alpha^s r^t r^s \left(\mathbf{x}^t\right)^T \mathbf{x}^s + \sum_t \alpha^t$$

subject to $\sum_t \alpha^t r^t = 0$ and $\alpha^t \geq 0, \forall t$

Most α^t are 0 and only a small number have $\alpha^t > 0$; they are the support vectors

Soft Margin Hyperplane

- Not linearly separable

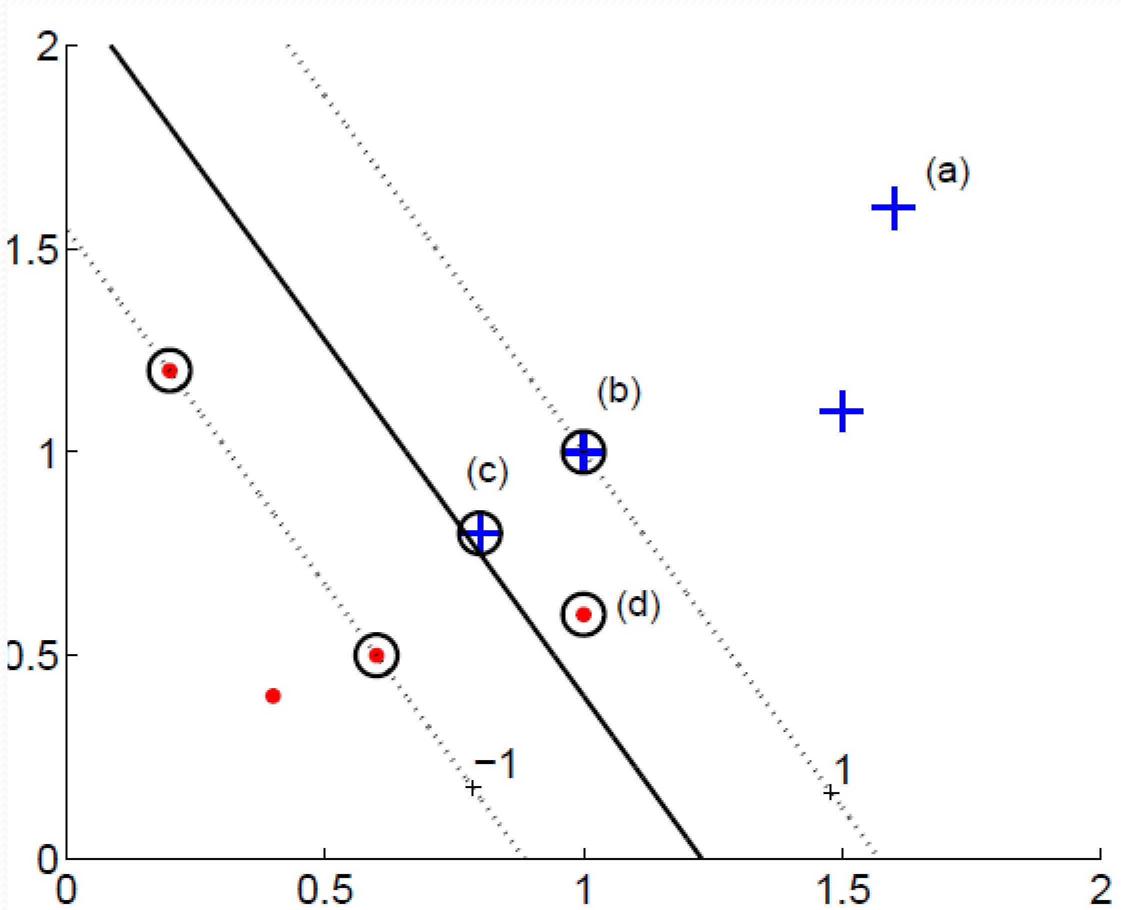
$$r^t\left(\mathbf{w}^T x^t + w_0\right) \geq 1 - \xi^t$$

- Soft error

$$\sum_t \xi^t$$

- New primal is

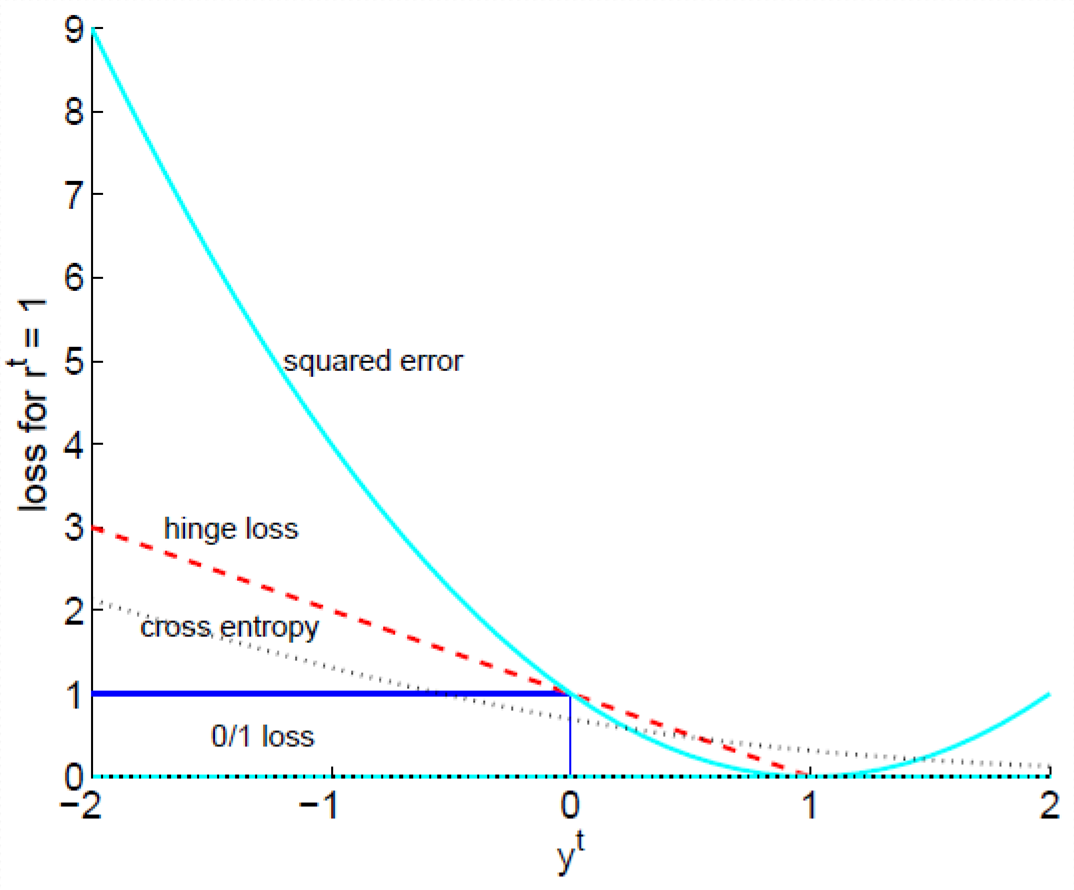
$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t \xi^t - \sum_t \alpha^t\left[r^t\left(\mathbf{w}^T x^t + w_0\right) - 1 + \xi^t\right] - \sum_t \mu^t \xi^t$$



(a)
(b)
(c)
(d)
−1
1
0
0.5
1
1.5
2

Hinge Loss

$$L_{hinge}(y^t, r^t) = \begin{cases} 0 & \text{if } y^t r^t \geq 1 \\ 1 - y^t r^t & \text{otherwise} \end{cases}$$



ν-SVM

$$\min \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{N}\sum_t \xi^t$$

subject to

$$r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq \rho - \xi^t, \xi^t \geq 0, \rho \geq 0$$

$$L_d = -\frac{1}{2}\sum_{t=1}^{N}\sum_s \alpha^t\alpha^s r^t r^s \left(x^t\right)^T x^s$$

subject to

$$\sum_{\mathrm{t}} \alpha^t r^t = 0, 0 \leq \alpha^t \leq \frac{1}{N}, \sum_t \alpha^t \geq \nu$$

ν controls the fraction of support vectors

Kernel Trick

- Preprocess input ***x*** by basis functions

$$\boldsymbol{z} = \boldsymbol{\varphi}(\boldsymbol{x}) \qquad g(\boldsymbol{z})=\boldsymbol{w}^T\boldsymbol{z}$$

$$g(\boldsymbol{x})=\boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x})$$

- The SVM solution

$$\mathbf{w} = \sum_t \alpha^t r^t \mathbf{z}^t = \sum_t \alpha^t r^t \boldsymbol{\varphi}\left(\mathbf{x}^t\right)$$

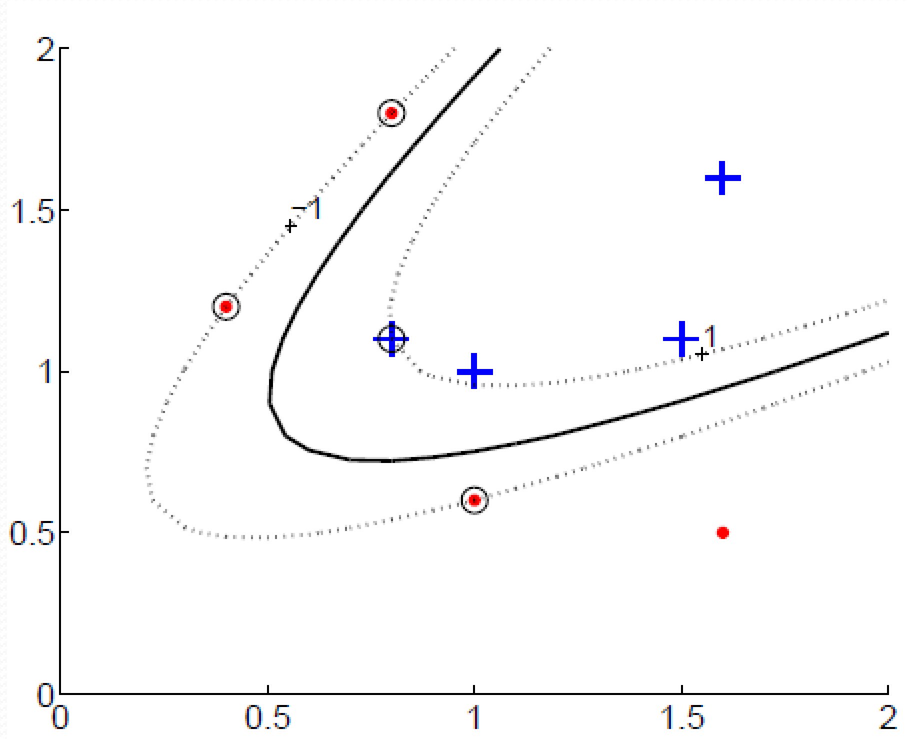
$$g(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = \sum_t \alpha^t r^t \boxed{\boldsymbol{\varphi}\left(\mathbf{x}^t\right)^T \boldsymbol{\varphi}(\mathbf{x})}$$

$$g(\mathbf{x}) = \sum_t \alpha^t r^t \boxed{K\left(\mathbf{x}^t, \mathbf{x}\right)}$$

Vectorial Kernels

- Polynomials of degree *q*:

$$K(\mathbf{x}^t, \mathbf{x}) = (\mathbf{x}^T \mathbf{x}^t + 1)^q$$



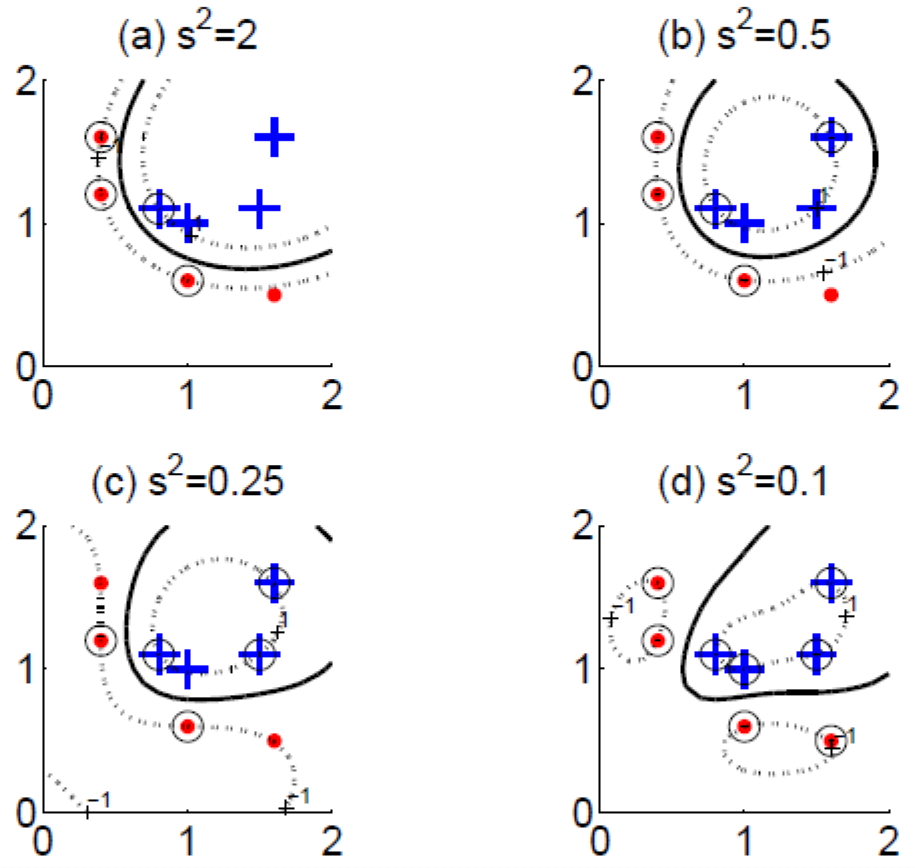
$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^2$$
$$= (x_1 y_1 + x_2 y_2 + 1)^2$$
$$= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2$$

$$\phi(\mathbf{x}) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2\right]^T$$

Vectorial Kernels

- Radial-basis functions:

$$K\left(\mathbf{x}^t,\mathbf{x}\right)=\exp\left[-\frac{\left\|\mathbf{x}^t-\mathbf{x}\right\|^2}{2s^2}\right]$$



Defining kernels

- Kernel "engineering"
- Defining good measures of similarity
- String kernels, graph kernels, image kernels, ...
- Empirical kernel map: Define a set of templates $\boldsymbol{m}_i$ and score function $s(\boldsymbol{x},\boldsymbol{m}_i)$

  $\phi(\boldsymbol{x}^t)=[s(\boldsymbol{x}^t,\boldsymbol{m}_1), s(\boldsymbol{x}^t,\boldsymbol{m}_2),\ldots, s(\boldsymbol{x}^t,\boldsymbol{m}_M)]$

  and

  $K(\boldsymbol{x},\boldsymbol{x}^t)=\phi(\boldsymbol{x})^T \phi(\boldsymbol{x}^t)$

Multiple Kernel Learning

- Fixed kernel combination

$$K(\mathbf{x},\mathbf{y}) = \begin{cases} cK(\mathbf{x},\mathbf{y}) \\ K_1(\mathbf{x},\mathbf{y}) + K_2(\mathbf{x},\mathbf{y}) \\ K_1(\mathbf{x},\mathbf{y})K_2(\mathbf{x},\mathbf{y}) \end{cases}$$

- Adaptive kernel combination

$$K(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \eta_i K_i(\mathbf{x},\mathbf{y})$$

$$L_d = \sum_t \alpha^t - \frac{1}{2}\sum_t \sum_s \alpha^t \alpha^s r^t r^s \sum_i \eta_i K_i\left(\mathbf{x}^t, \mathbf{x}^s\right)$$

$$g(\mathbf{x}) = \sum_t \alpha^t r^t \sum_i \eta_i K_i\left(\mathbf{x}^t, \mathbf{x}\right)$$

- Localized kernel combination

$$g(\mathbf{x}) = \sum_t \alpha^t r^t \sum_i \eta_i(\mathbf{x} \mid \theta) K_i\left(\mathbf{x}^t, \mathbf{x}\right)$$

Multiclass Kernel Machines

- 1-vs-all
- Pairwise separation
- Error-Correcting Output Codes (section 17.5)
- Single multiclass optimization

$$\min \frac{1}{2}\sum_{i=1}^{K}\|\mathbf{w}_i\|^2 + C\sum_i\sum_t \xi_i^t$$

subject to

$$\mathbf{w}_{z^t}{}^T\mathbf{x}^t + w_{z^t 0} \geq \mathbf{w}_i{}^T\mathbf{x}^t + w_{i0} + 2 - \xi_i^t, \forall i \neq z^t, \xi_i^t \geq 0$$

SVM for Regression

- Use a linear model (possibly kernelized)

$$f(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0$$

- Use the ϵ-sensitive error function

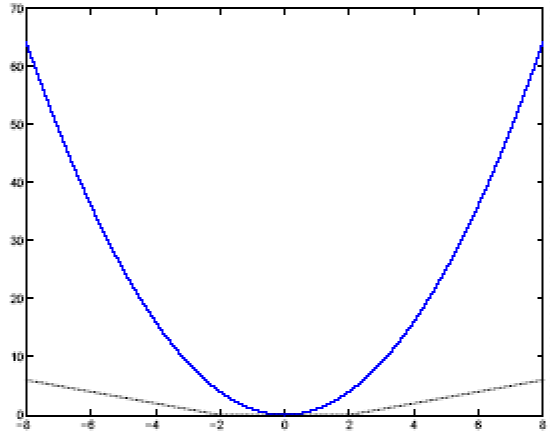
$$e_\varepsilon\left(r^t, f\left(\mathbf{x}^t\right)\right)=\begin{cases}0 & \text{if } \left|r^t-f\left(\mathbf{x}^t\right)\right|<\varepsilon \\ \left|r^t-f\left(\mathbf{x}^t\right)\right|-\varepsilon & \text{otherwise}\end{cases}$$

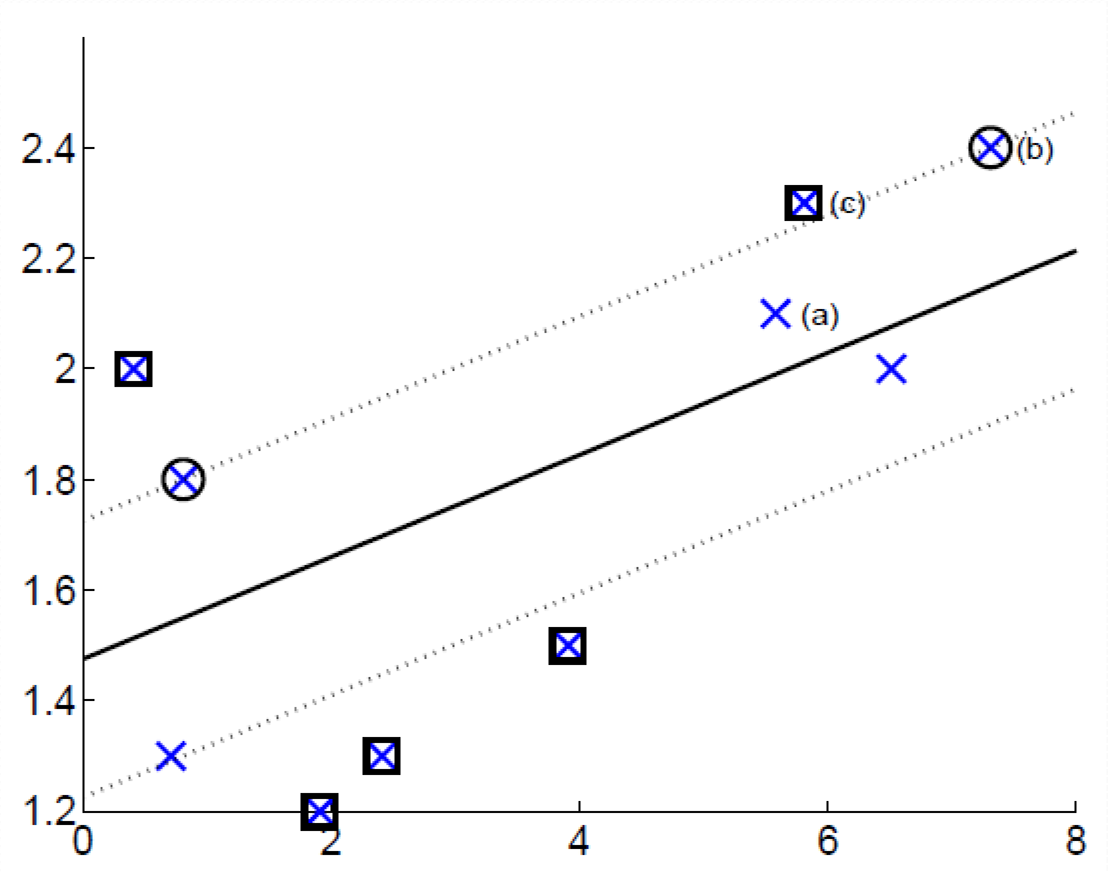
- $$\min \frac{1}{2}\|\mathbf{w}\|^2+C\sum_t\left(\xi_+^t+\xi_-^t\right)$$

$$r^t-\left(\mathbf{w}^T\mathbf{x}+w_0\right)\leq\varepsilon+\xi_+^t$$

$$\left(\mathbf{w}^T\mathbf{x}+w_0\right)-r^t\leq\varepsilon+\xi_-^t$$

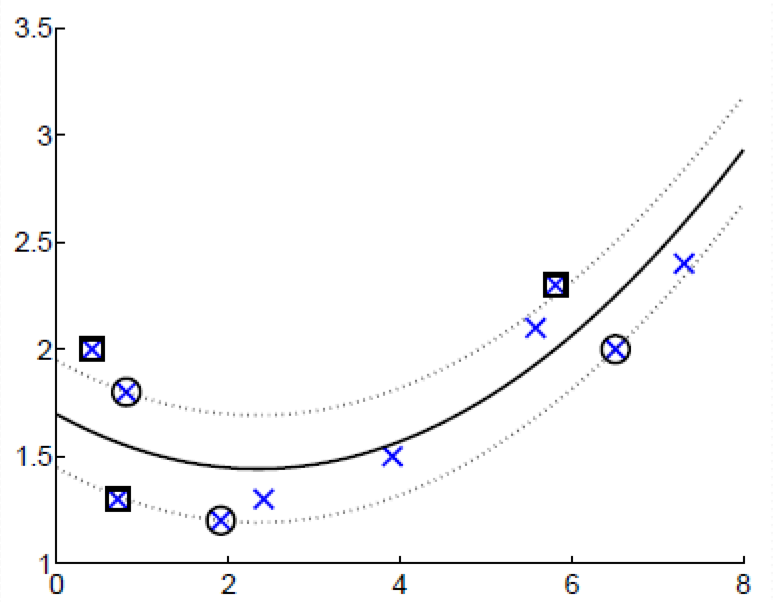
$$\xi_+^t,\xi_-^t\geq 0$$



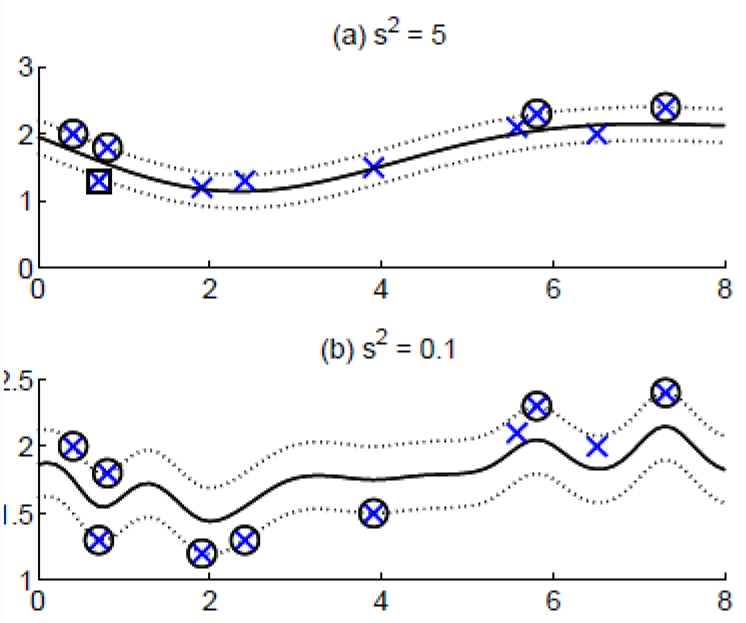

2.4
2.2
2
1.8
1.6
1.4
1.2
0
2
4
6
8
(a)
(b)
(c)

Kernel Regression

- Polynomial kernel



- Gaussian kernel



One-Class Kernel Machines

- Consider a sphere with center ***a*** and radius *R*

$$\min R^2 + C\sum_t \xi^t$$

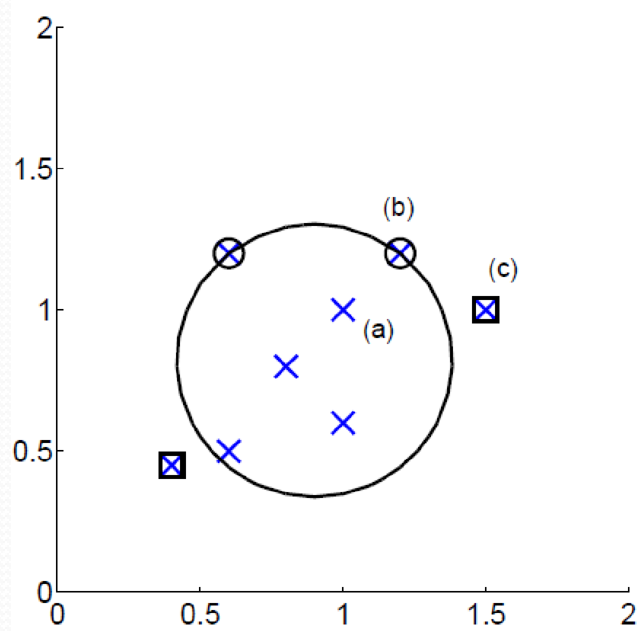
subject to

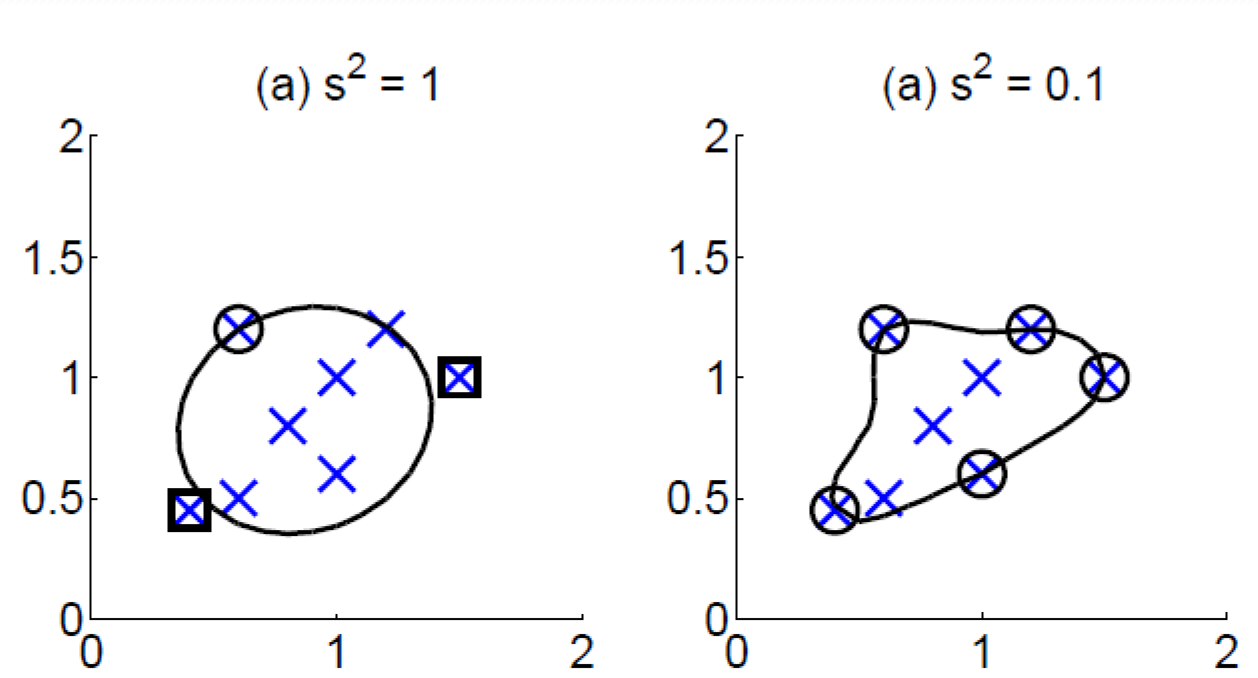
$$\left\|\mathbf{x}^t - a\right\| \le R^2 + \xi^t, \xi^t \ge 0$$

$$L_d = \sum_t \alpha^t \boxed{\left(x^t\right)^T x^s} - \sum_{t=1}^{N}\sum_s \alpha^t \alpha^s r^t r^s \boxed{\left(x^t\right)^T x^s}$$

subject to

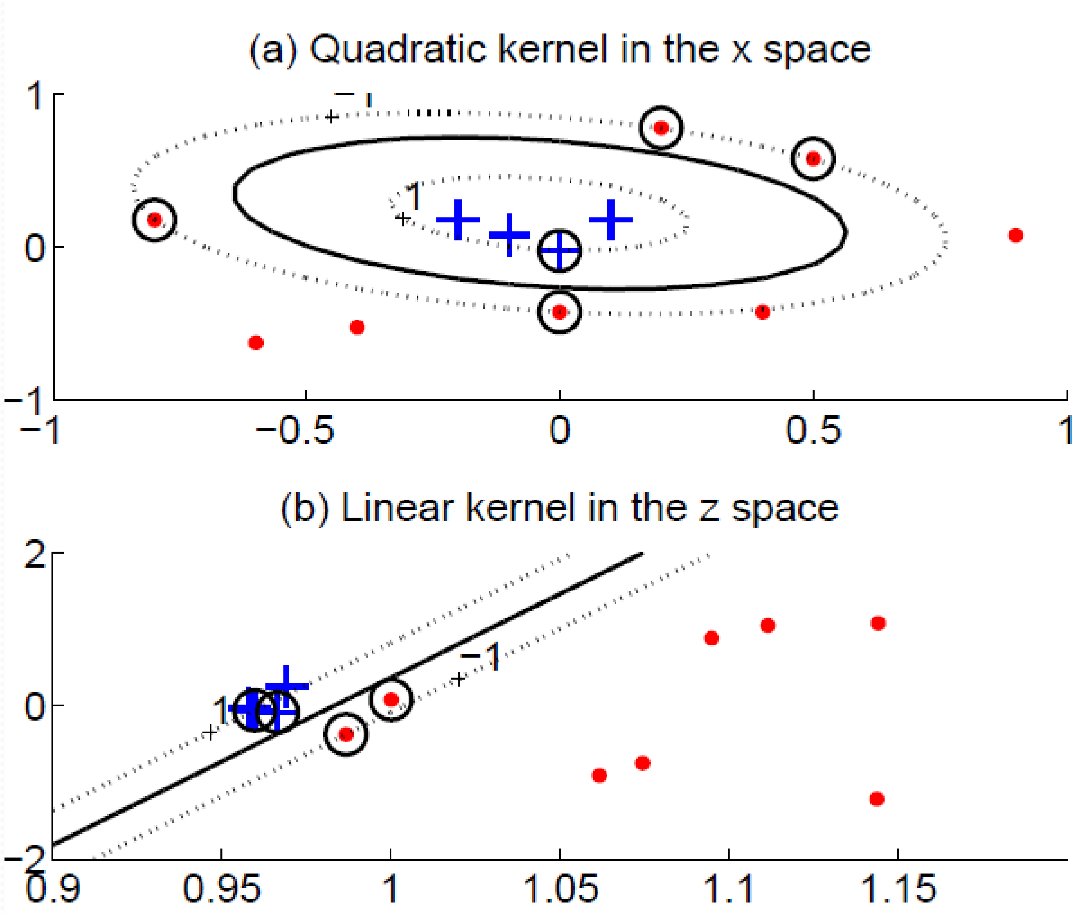
$$0 \le \alpha^t \le C, \sum_t \alpha^t = 1$$




(a) s2 = 1
(a) s2 = 0.1
2
1.5
1
0.5
0
0
1
2

Kernel Dimensionality Reduction

- Kernel PCA does PCA on the kernel matrix (equal to canonical PCA with a linear kernel)
- Kernel LDA



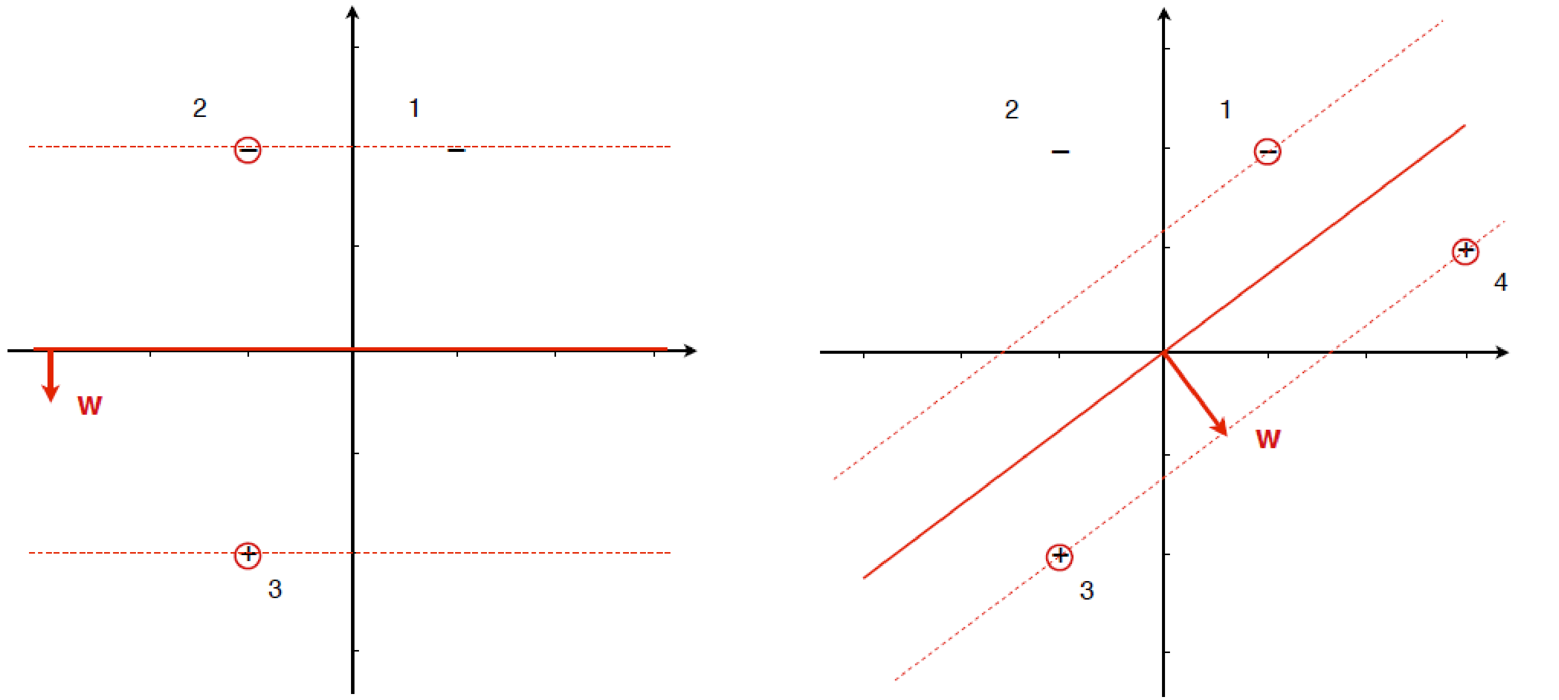
Numerical Example

The dual optimisation problem for support vector machines is to maximise the dual Lagrangian under positivity constraints and one equality constraint:

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\arg\max} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

- This example and following slides are obtained from «**Machine Learning The Art and Science of Algorithms that Make Sense of Data**» **book.**
- http://people.cs.bris.ac.uk/~flach/mlbook//



(left) A maximum-margin classifier built from three examples, with $\mathbf{w} = (0, -1/2)$ and margin 2. The circled examples are the support vectors: they receive non-zero Lagrange multipliers and define the decision boundary. **(right)** By adding a second positive the decision boundary is rotated to $\mathbf{w} = (3/5, -4/5)$ and the margin decreases to 1.

- From «**Machine Learning The Art and Science of Algorithms that Make Sense of Data» book.** http://people.cs.bris.ac.uk/~flach/mlbook//

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ -1 & 2 \\ -1 & -2 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \end{pmatrix}$$

The matrix $\mathbf{X}'$ on the right incorporates the class labels; i.e., the rows are $y_i \mathbf{x}_i$. The Gram matrix is (without and with class labels):

$$\mathbf{X}\mathbf{X}^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & -5 \\ 3 & 5 & -3 \\ -5 & -3 & 5 \end{pmatrix} \quad \mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & 5 \\ 3 & 5 & 3 \\ 5 & 3 & 5 \end{pmatrix}$$

The dual optimisation problem is thus

$$\begin{aligned} &\mathop{\arg\max}_{\alpha_1,\alpha_2,\alpha_3} -\frac{1}{2}\left(5\alpha_1^2 + 3\alpha_1\alpha_2 + 5\alpha_1\alpha_3 + 3\alpha_2\alpha_1 + 5\alpha_2^2 + 3\alpha_2\alpha_3 + 5\alpha_3\alpha_1 + 3\alpha_3\alpha_2 + 5\alpha_3^2\right) + \alpha_1 + \alpha_2 + \alpha_3 \\ = &\mathop{\arg\max}_{\alpha_1,\alpha_2,\alpha_3} -\frac{1}{2}\left(5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 + 5\alpha_2^2 + 6\alpha_2\alpha_3 + 5\alpha_3^2\right) + \alpha_1 + \alpha_2 + \alpha_3 \end{aligned}$$

subject to $\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$ and $-\alpha_1 - \alpha_2 + \alpha_3 = 0$.

- From «**Machine Learning The Art and Science of Algorithms that Make Sense of Data» book.** http://people.cs.bris.ac.uk/~flach/mlbook//

- Using the equality constraint we can eliminate one of the variables, say α_3, and simplify the objective function to

$$\underset{\alpha_1,\alpha_2,\alpha_3}{\arg\max} -\frac{1}{2}\left(20\alpha_1^2 + 32\alpha_1\alpha_2 + 16\alpha_2^2\right) + 2\alpha_1 + 2\alpha_2$$

- Setting partial derivatives to 0 we obtain $-20\alpha_1 - 16\alpha_2 + 2 = 0$ and $-16\alpha_1 - 16\alpha_2 + 2 = 0$ (notice that, because the objective function is quadratic, these equations are guaranteed to be linear).
- We therefore obtain the solution $\alpha_1 = 0$ and $\alpha_2 = \alpha_3 = 1/8$. We then have $\mathbf{w} = 1/8(\mathbf{x}_3 - \mathbf{x}_2) = \begin{pmatrix} 0 \\ -1/2 \end{pmatrix}$, resulting in a margin of $1/||\mathbf{w}|| = 2$.
- Finally, t can be obtained from any support vector, say $\mathbf{x}_2$, since $y_2(\mathbf{w}\cdot\mathbf{x}_2 - t) = 1$; this gives $-1\cdot(-1-t) = 1$, hence $t = 0$.

- From «**Machine Learning The Art and Science of Algorithms that Make Sense of Data» book.** http://people.cs.bris.ac.uk/~flach/mlbook//

We now add an additional positive at $(3,1)$. This gives the following data matrices:

$$\mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \\ 3 & 1 \end{pmatrix} \qquad \mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & 5 & -5 \\ 3 & 5 & 3 & 1 \\ 5 & 3 & 5 & -5 \\ -5 & 1 & -5 & 10 \end{pmatrix}$$

- It can be verified by similar calculations to those above that the margin decreases to 1 and the decision boundary rotates to $\mathbf{w} = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix}$.
- The Lagrange multipliers now are $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$ and $\alpha_4 = 2/5$. Thus, only $\mathbf{x}_3$ is a support vector in both the original and the extended data set.

- From «**Machine Learning The Art and Science of Algorithms that Make Sense of Data» book.** http://people.cs.bris.ac.uk/~flach/mlbook//

SVM with Hinge Loss

Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

The constraint $y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

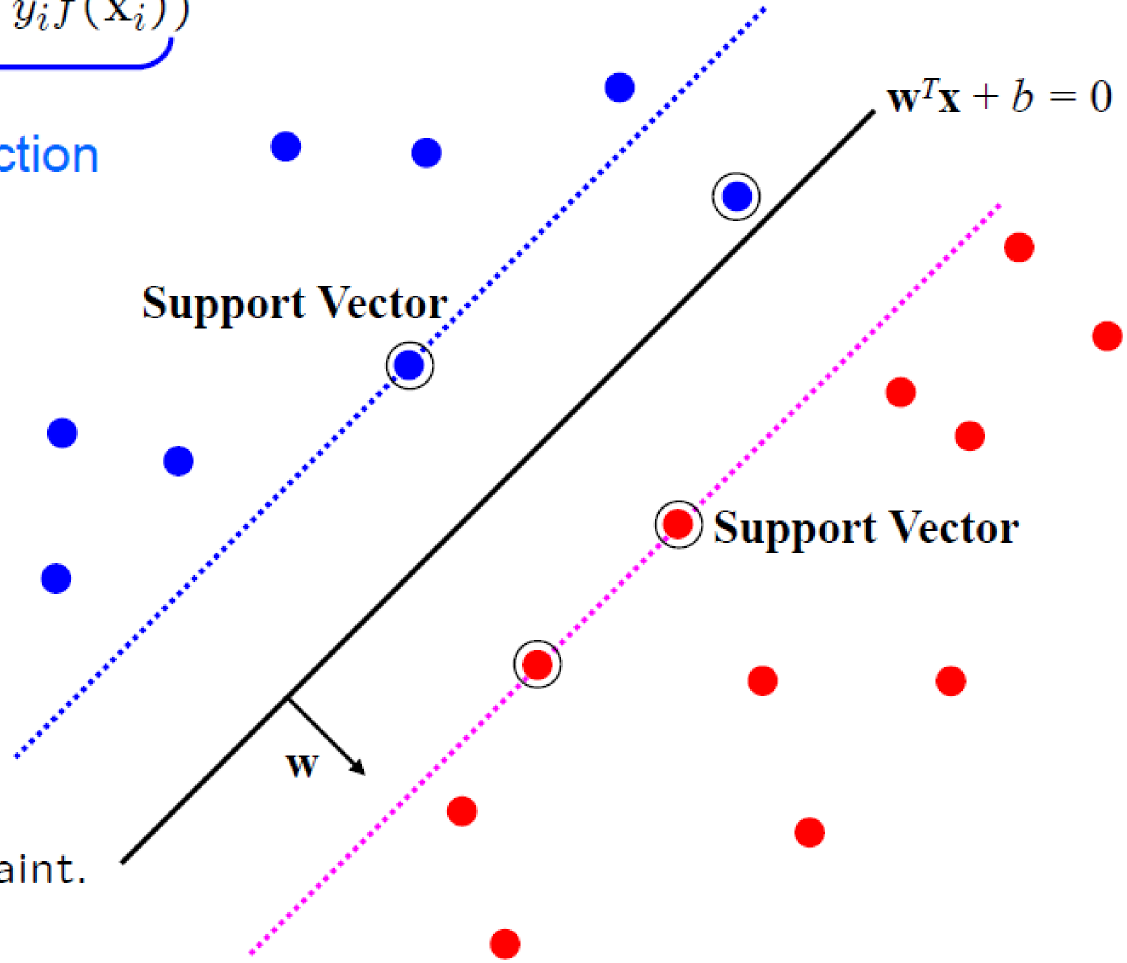
- Slides are adopted from A. Zisserman

Loss function

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

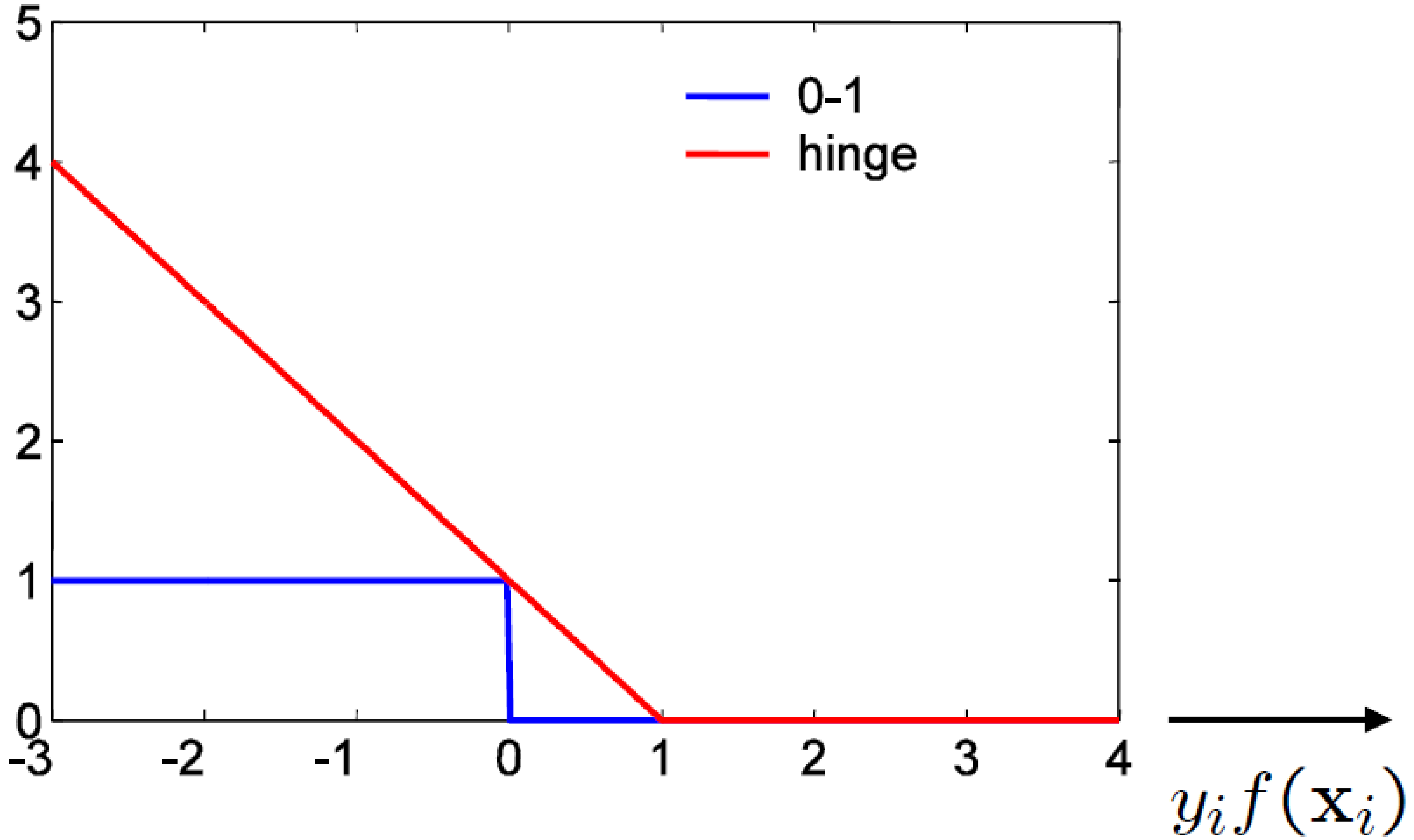
Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss



- Adopted from A. Zisserman

Loss functions

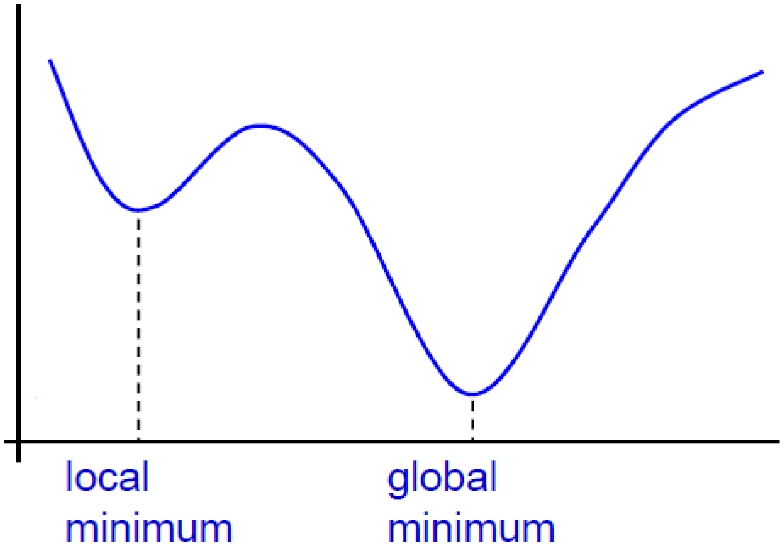


- SVM uses "hinge" loss $\max(0, 1 - y_i f(\mathbf{x}_i))$
- an approximation to the 0-1 loss

- Adopted from A. Zisserman

Optimization continued

$$\min_{\mathbf{w}\in\mathbb{R}^d} C\sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2$$



- Does this cost function have a unique solution?
- Does the solution depend on the starting point of an iterative optimization algorithm (such as gradient descent)?

If the cost function is convex, then a locally optimal point is globally optimal (provided the optimization is over a convex set, which it is in our case)

- Adopted from A. Zisserman

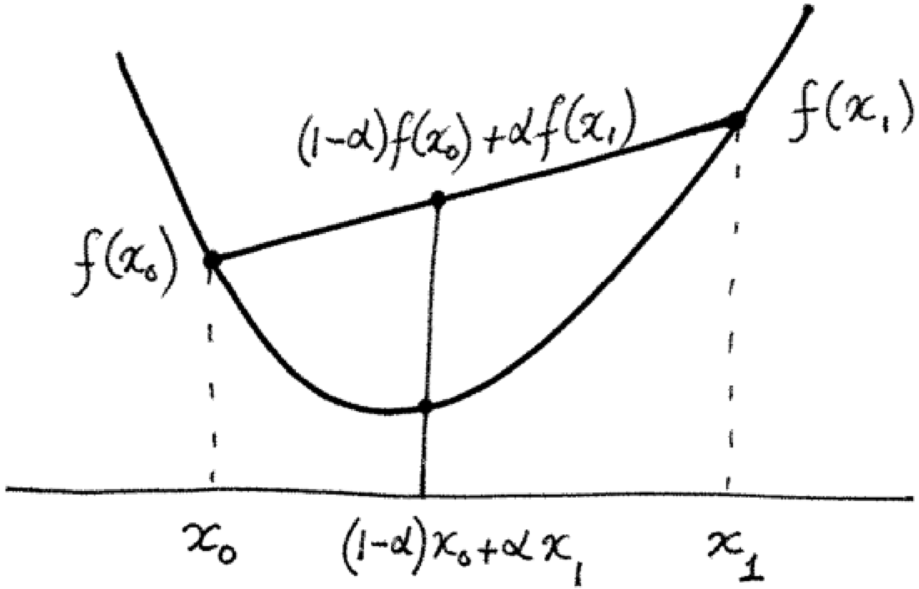
Convex functions

D – a domain in $\mathbb{R}^n$.

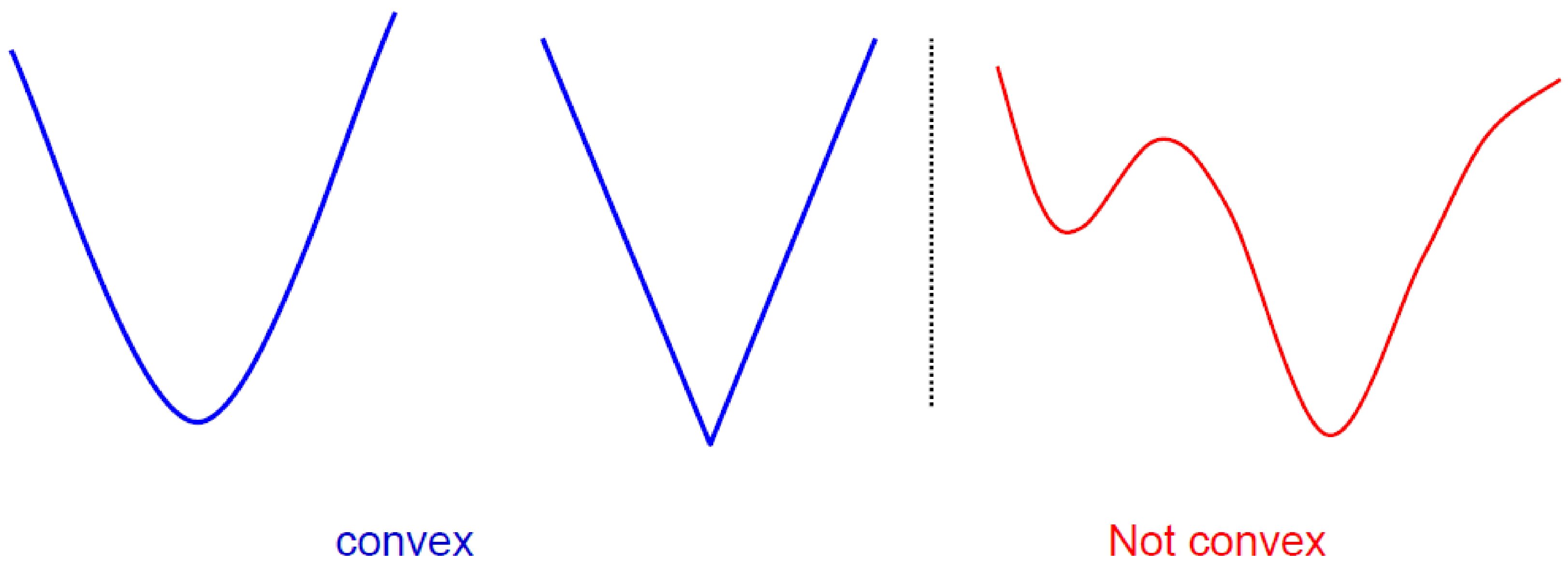
A convex function $f : D \to \mathbb{R}$ is one that satisfies, for any $\mathbf{x}_0$ and $\mathbf{x}_1$ in D:

$$f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{x}_1) \leq (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_1) \ .$$

Line joining $(\mathbf{x}_0, f(\mathbf{x}_0))$ and $(\mathbf{x}_1, f(\mathbf{x}_1))$ lies above the function graph.

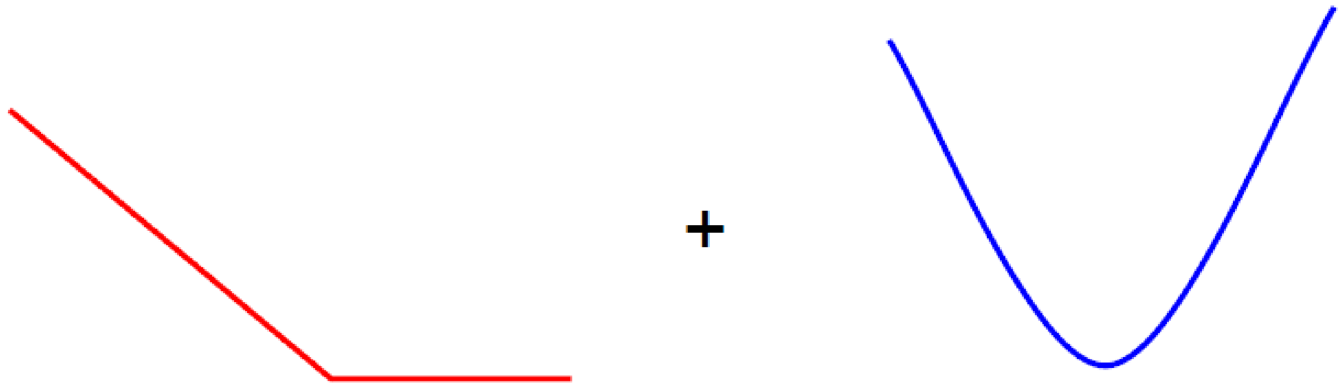


Convex function examples



A non-negative sum of convex functions is convex

- Adopted from A. Zisserman



SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} C \sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2 \qquad \text{convex}$$

- Adopted from A. Zisserman

Gradient (or steepest) descent algorithm for SVM

To minimize a cost function $\mathcal{C}(\mathbf{w})$ use the iterative update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \mathcal{C}(\mathbf{w}_t)$$

where η is the learning rate.

First, rewrite the optimization problem as an average

$$\begin{aligned}
\min_{\mathbf{w}} \mathcal{C}(\mathbf{w}) &= \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) \\
&= \frac{1}{N}\sum_i^N \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \max(0, 1 - y_i f(\mathbf{x}_i)) \right)
\end{aligned}$$

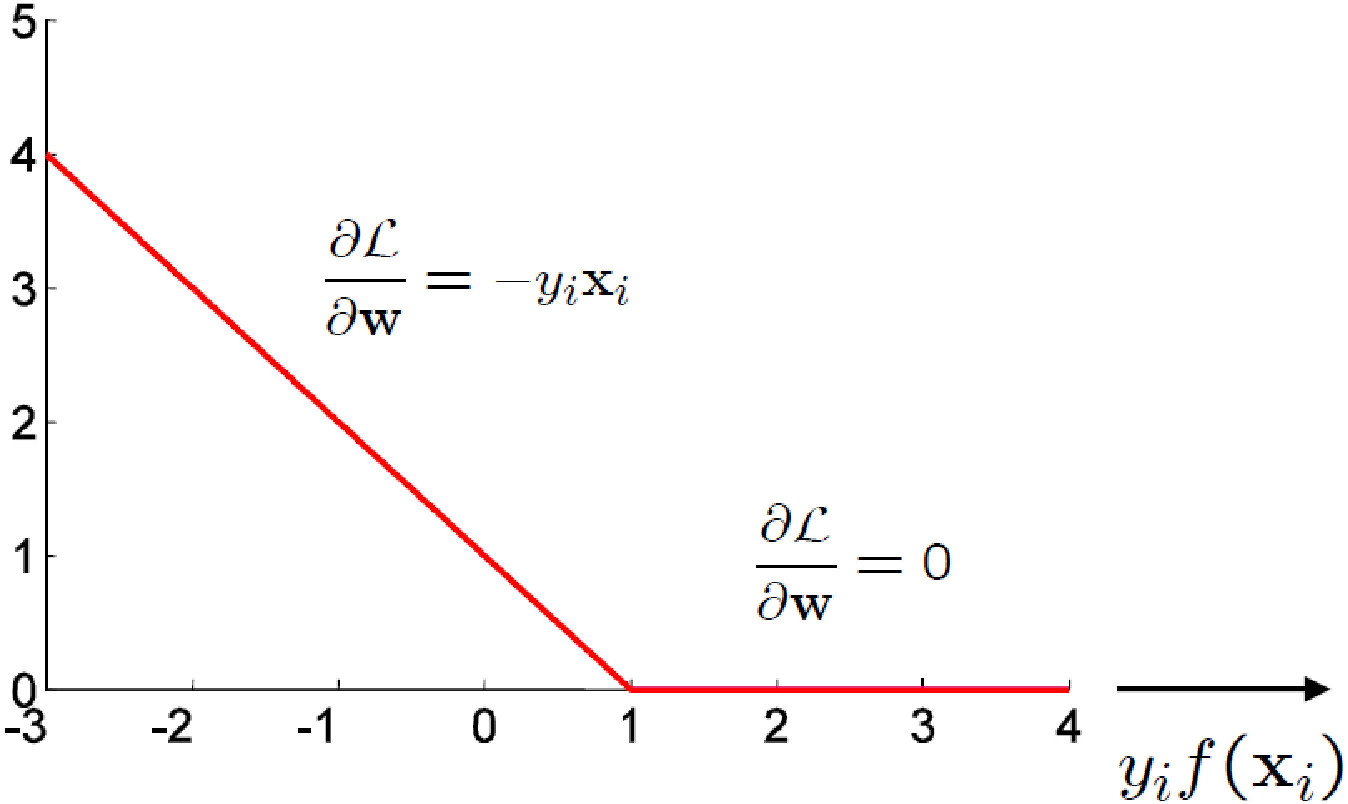
(with $\lambda = 2/(NC)$ up to an overall scale of the problem) and $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$

Because the hinge loss is not differentiable, a sub-gradient is computed

- Adopted from A. Zisserman

Sub-gradient for hinge loss

$$\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f(\mathbf{x}_i)) \qquad f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$



- Adopted from A. Zisserman

Sub-gradient descent algorithm for SVM

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N}\sum_{i}^{N}\left(\frac{\lambda}{2}||\mathbf{w}||^2 + \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})\right)$$

The iterative update is

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta\nabla_{\mathbf{w}_t}\mathcal{C}(\mathbf{w}_t) \\ &\leftarrow \mathbf{w}_t - \eta\frac{1}{N}\sum_{i}^{N}(\lambda\mathbf{w}_t + \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}_t)) \end{aligned}$$

where η is the learning rate.

Then each iteration t involves cycling through the training data with the updates:

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta(\lambda\mathbf{w}_t - y_i\mathbf{x}_i) && \text{if } y_i f(\mathbf{x}_i) < 1 \\ &\leftarrow \mathbf{w}_t - \eta\lambda\mathbf{w}_t && \text{otherwise} \end{aligned}$$

In the Pegasos algorithm the learning rate is set at $\eta_t = \frac{1}{\lambda t}$

- Adopted from A. Zisserman