annotated version

Machine Learning Course - CS-433

# Support Vector Machines

Oct 29, 2019

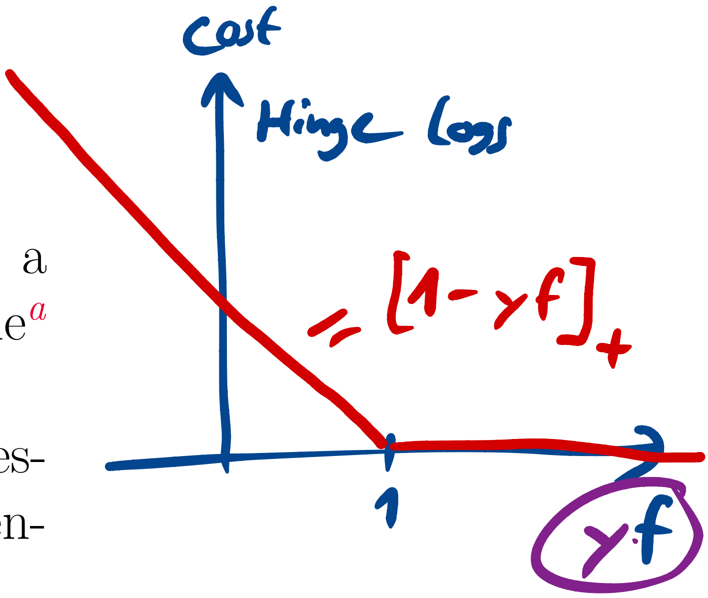
changes by Martin Jaggi 2019, changes by Rüdiger Urbanke 2018, changes by Martin Jaggi 2016, 2017 

Last updated on: October 29, 2019

EPFL

# Motivation

By changing the cost function of a linear classifier from Logistic to Hinge, we obtain the support vector machine (SVM).




# Support Vector Machine

Throughout, we will work with a classification problem and assume[a] that the labels $y_n \in \{\pm 1\}$.
(This is in contrast to logistic regression, where we have used the convention $y_n \in \{0, 1\}$.)
We again write $\mathbf{x}_n$ for datapoint $n$, and assume that all constructed features and a potential constant bias are already included in $\mathbf{x}_n$.

$f = x_n^\top w$

The SVM optimizes the following cost:

label $y_n \in \{+1, -1\}$

$$\min_{\mathbf{w}} \sum_{n=1}^{N} [1 - \underbrace{y_n}_{\text{label}} \underbrace{\mathbf{x}_n^\top \mathbf{w}}_{f}]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

where the first term is the Hinge loss defined as $[z]_+ := \max\{0, z\}$.

[a]Note that for any use-case, the labels can be converted accordingly before training, and after prediction.

# Hinge vs MSE vs Logistic

Consider $y \in \{-1, +1\}$ with prediction $f \in \mathbb{R}$, then the three cost functions can be written as follows:

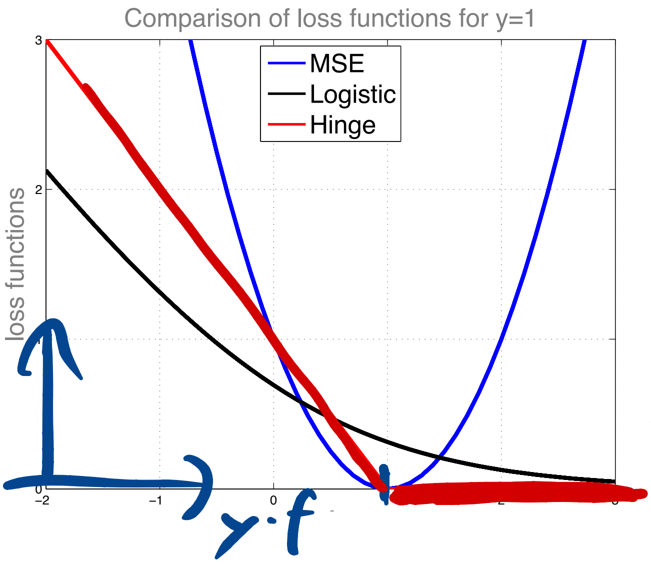
$$Hinge(f) = [1 - yf]_+$$
$$MSE(f) = (1 - yf)^2$$
$$logisticLoss(f) = \log\left(1 + e^{-yf}\right)$$

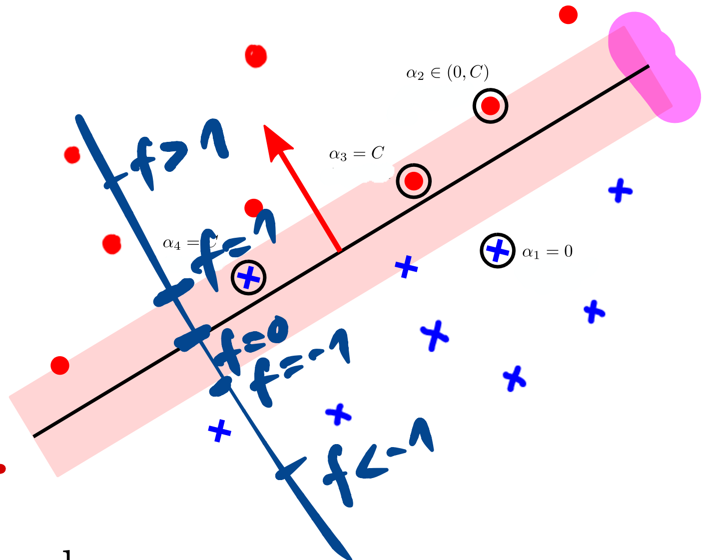
SVM

Regression for Classification

Logistic Regression

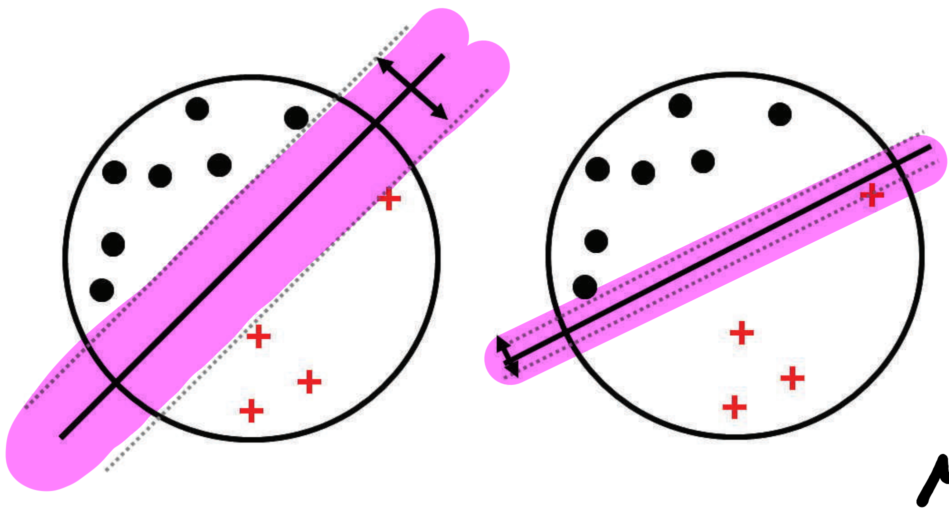
Homework y transform




Notice the margin in the Hinge loss. SVM is a maximum margin method.

Assumption: linearly separa



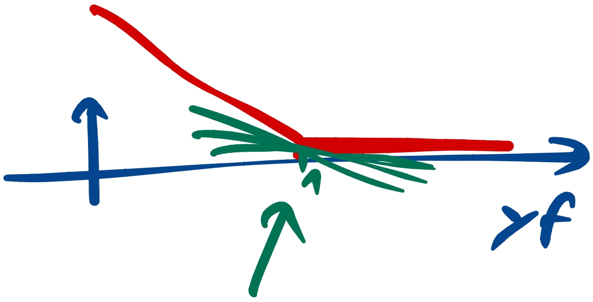
Margin $\approx 1/\|w\|$

(not proven)

## Optimization

in $\mathbf{w}$ : yes

Is this function convex? Is it differentiable? (in $\mathbf{w}$) no !




$$\min_{\mathbf{w}} \sum_{n=1}^{N} \Big[1 - y_n \overbrace{\mathbf{x}_n^\top \mathbf{w}}^{f}\Big]_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Can use SGD! (with subgradients).

Is there a better optimization algorithm here?

## Duality: The big picture

Let us say that we are interested in optimizing a function $\mathcal{L}(\mathbf{w})$ and it is a difficult problem. Define an auxiliary function $G(\mathbf{w}, \boldsymbol{\alpha})$ such that

$$\mathcal{L}(\mathbf{w}) = \max_{\boldsymbol{\alpha}} \; G(\mathbf{w}, \boldsymbol{\alpha}).$$

so that we can then choose between optimizing either of

$$\min_{\boldsymbol{w}} \overbrace{\max_{\boldsymbol{\alpha}} G(\mathbf{w}, \boldsymbol{\alpha})}^{\mathcal{L}(w)} = \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}} G(\mathbf{w}, \boldsymbol{\alpha})$$

Primal problem Dual problem

Three questions:

1. How do you set $G(\mathbf{w}, \boldsymbol{\alpha})$?
2. When is it OK to switch $\min_{\boldsymbol{w}}$ and $\max_{\boldsymbol{\alpha}}$ ?
3. When is the dual easier to optimize than the primal?

**Q1:** How to obtain $G(\mathbf{w}, \boldsymbol{\alpha})$?

For one datapoint

$$[v_n]_+ := \max\{0, v_n\}$$

$$\underset{\Delta}{=} \max_{\alpha_n} \alpha_n v_n \text{ where } \alpha_n \in [0,1]$$

$$\underbrace{[1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+}_{\text{hinge-loss}} = \max_{\alpha_n \in [0,1]} \alpha_n \underbrace{(1 - y_n \mathbf{x}_n^\top \mathbf{w})}_{v_n}$$

proof:

- case $v \geq 0$
  max $= v$
  attained when $\alpha = 1$
- case $v < 0$
  max $= 0$
  attaind at $\alpha = 0$

For all points:

We can rewrite the SVM problem as:

$$\min_{\boldsymbol{w}} \underbrace{\max_{\boldsymbol{\alpha} \in [0,1]^N} \underbrace{\sum_{n=1}^{N} \alpha_n (1 - y_n \mathbf{x}_n^\top \mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2}_{=: \, G(\boldsymbol{w}, \boldsymbol{\alpha})}}_{\mathcal{L}(w) \, = \max_{\alpha} G(w, \alpha)}$$

This is differentiable, convex in $\mathbf{w}$ and concave in $\boldsymbol{\alpha}$.

over both $w$ and $\alpha$

**Q2:** When is it OK to switch max and min? Using a minimax theorem, it is OK to do so when $G(\mathbf{w}, \boldsymbol{\alpha})$ is convex in $\mathbf{w}$ and concave in $\boldsymbol{\alpha}$, under weak additional assumptions.

always $\leq$

proof

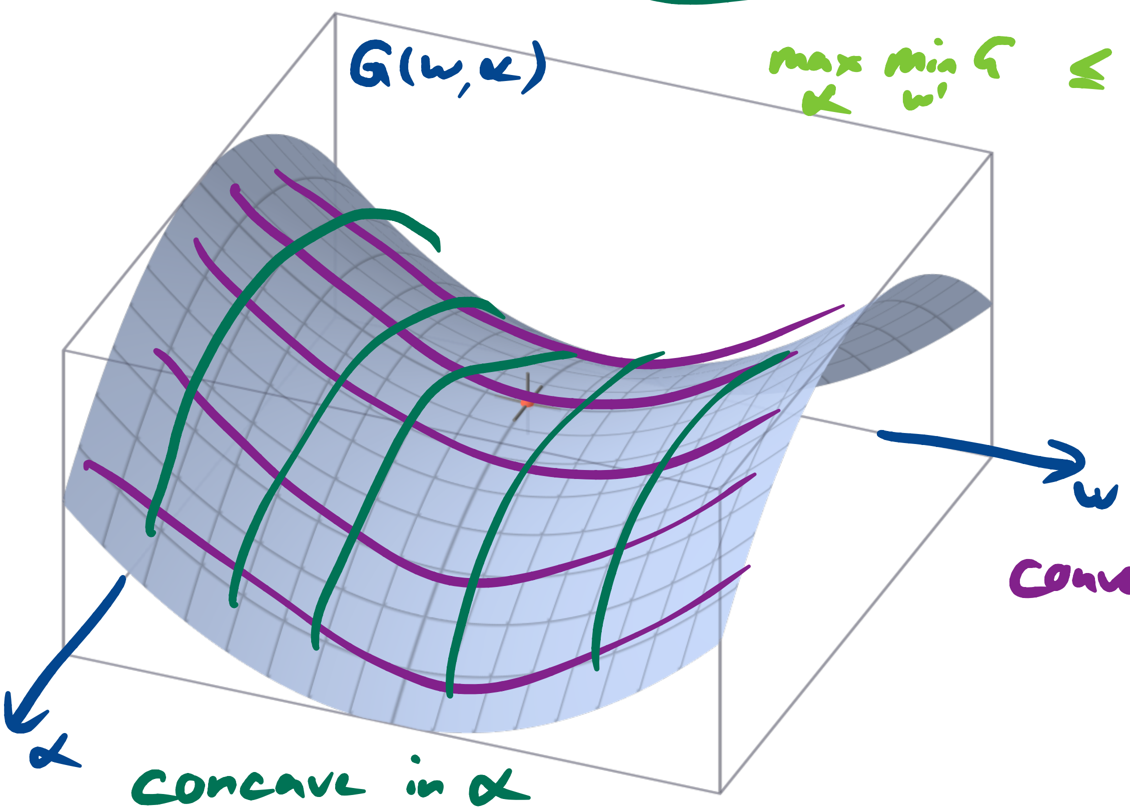
$\forall w, \alpha$

$$\min_{w'} G(w', \alpha) \leq G(w, \alpha)$$

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}} G(\mathbf{w}, \boldsymbol{\alpha}) = \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}} G(\mathbf{w}, \boldsymbol{\alpha})$$

$$\max_{\alpha} \min_{w'} G(w', \alpha) \leq \max_{\alpha} G(w, \alpha) \quad \forall w$$

$$\max_{\alpha} \min_{w'} G \leq \min_{w} \max_{\alpha} G(w, \alpha)$$




convex in w

For a more systematic way to derive suitable $G(\mathbf{w}, \boldsymbol{\alpha})$ and dual variables $\boldsymbol{\alpha}$, see the concept of convex conjugate functions, as in the language of Fenchel duality.
See e.g. Bertsekas' "Nonlinear Programming" for more formal details.

For SVM, switching the min and max, we have the following saddle-point formulation

$$\max_{\boldsymbol{\alpha}\in[0,1]^N} \min_{\boldsymbol{w}} \sum_{n=1}^{N} \alpha_n(1 - y_n \mathbf{x}_n^\top \mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1}$$

fixed

$G(w, \alpha)$

Taking the derivative w.r.t. $\mathbf{w}$:

$$\nabla_{\boldsymbol{w}} G(\mathbf{w}, \boldsymbol{\alpha}) = -\sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n + \lambda \mathbf{w} \overset{!}{=} 0$$

Equating this to $\mathbf{0}$ (which is called the first-order optimality condition for $\mathbf{w}$), we have the correspondence

$$\Leftrightarrow w = \frac{1}{\lambda} \sum_{n=1}^{N} \alpha_n y_n x_n$$

$$\mathbf{w}(\boldsymbol{\alpha}) = \frac{1}{\lambda} \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = \frac{1}{\lambda} \mathbf{X}^\top \mathbf{Y} \boldsymbol{\alpha}$$

where $\mathbf{Y} := \operatorname{diag}(\mathbf{y})$, and $\mathbf{X}$ again collects all $N$ data examples as its rows.

Plugging this $\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$ back into the saddle-point formulation (1), we have the dual optimization problem:

$$\max_{\boldsymbol{\alpha}\in[0,1]^N} \sum_{n=1}^{N} \alpha_n(1 - \frac{1}{\lambda} y_n \mathbf{x}_n^\top \mathbf{X}^\top \mathbf{Y} \boldsymbol{\alpha}) + \frac{\lambda}{2}\|\frac{1}{\lambda}\mathbf{X}^\top \mathbf{Y} \boldsymbol{\alpha}\|^2$$

$$= \max_{\boldsymbol{\alpha}\in[0,1]^N} \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2\lambda} \boldsymbol{\alpha}^\top \underbrace{\mathbf{Y}\mathbf{X}\mathbf{X}^\top\mathbf{Y}}_{\text{matrix}} \boldsymbol{\alpha}$$

Dual problem of SVM

**Q3:** When is the dual easier to optimize than the primal, and why?

(1) The dual is a differentiable (but constrained) quadratic problem.

$$\max_{\boldsymbol{\alpha}\in[0,1]^N} \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \mathbf{Q}\boldsymbol{\alpha},$$

where $\mathbf{Q} := \mathrm{diag}(\mathbf{y})\mathbf{X}\mathbf{X}^\top\mathrm{diag}(\mathbf{y})$. Optimization is easy by using co-ordinate descent, or more precisely coordinate ascent since this is a maximization problem. Crucially, this method will be changing only one $\alpha_n$ variable a time.

(2) The dual is naturally kernelized (just like the kernelized ridge, see next lecture) with $\mathbf{K} := \mathbf{X}\mathbf{X}^\top$.

(3) The solution $\boldsymbol{\alpha}$ is typically sparse, and is non-zero only for the training examples that are instrumental in determining the decision boundary.

Recall that $\alpha_n$ is the slope of lines that are lower bounds to the Hinge loss.

$$[1 - y_n f_n]_+ = \max_{\alpha_n \in [0,1]} \alpha_n (1 - y_n f_n)$$

There are 3 kinds of data vectors $\mathbf{x}_n$.

$yf > 1$

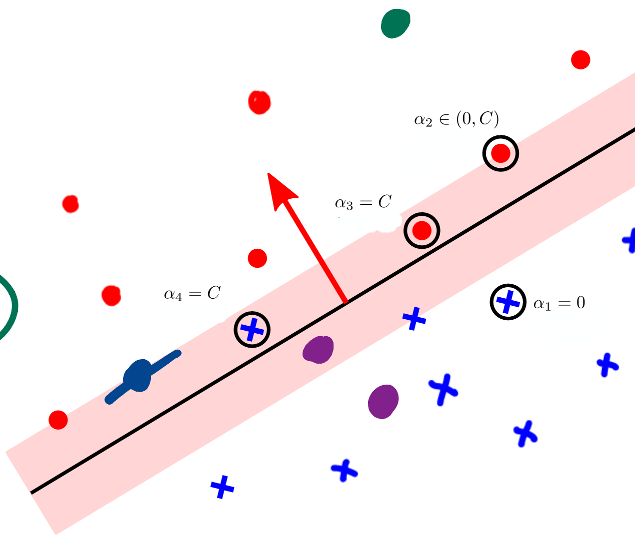
1. Non-support vectors. Examples that lie on the correct side outside the margin, so $\alpha_n = 0$.

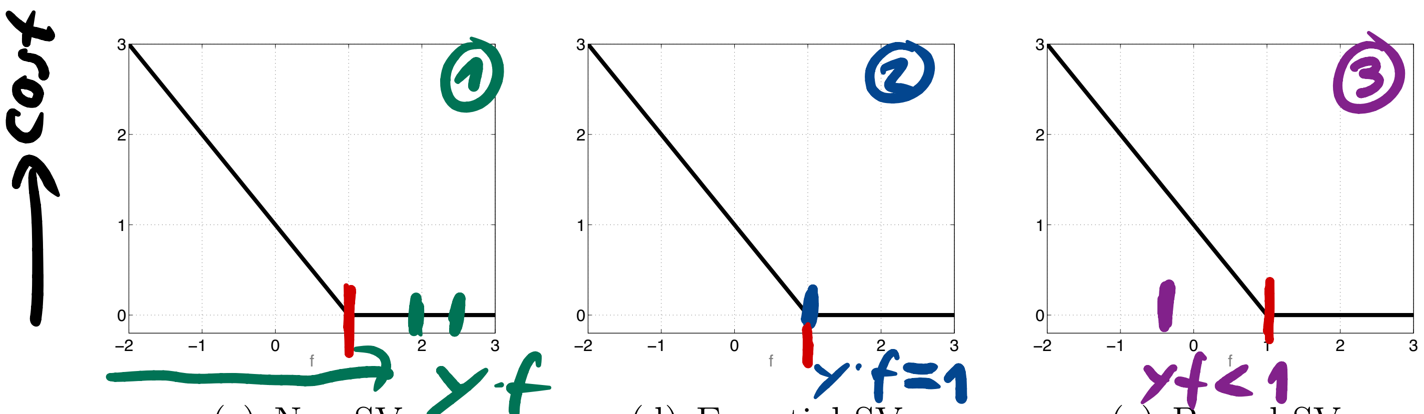
$yf = 1$

2. Essential support vectors. Examples that lie just on the margin, therefore $\alpha_n \in (0, 1)$.

$yf < 1$

3. Bound support vectors. Examples that lie strictly inside the margin, or on the wrong side, therefore $\alpha_n = 1$.



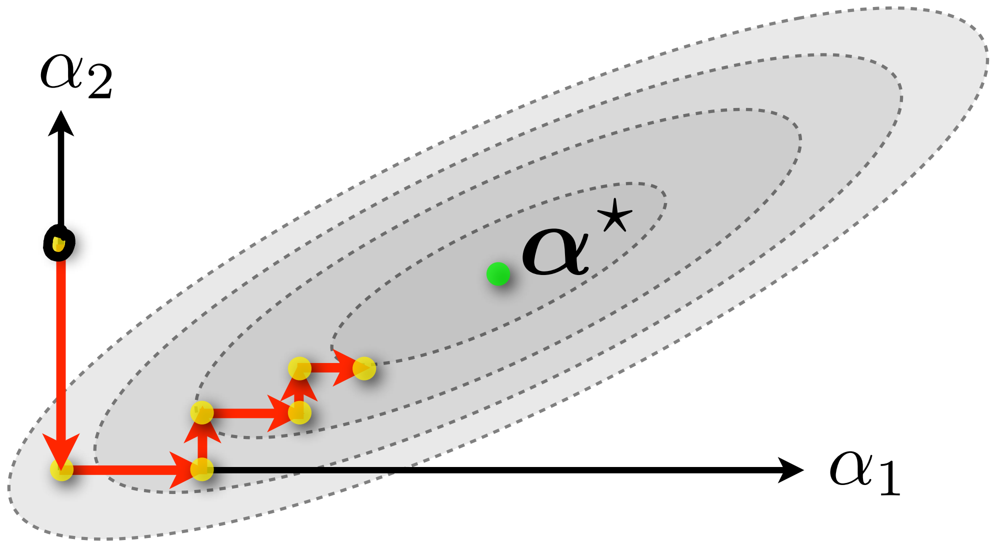




(c) Non-SV (d) Essential SV (e) Bound SV

## Coordinate Descent

**Goal:** Find $\boldsymbol{\alpha}^\star \in \mathbb{R}^N$ maximizing or minimizing $g(\boldsymbol{\alpha})$.
Yet another optimization algorithm?

**Idea:** Update one coordinate at a time, while keeping others fixed.




initialize $\boldsymbol{\alpha}^{(0)} \in \mathbb{R}^N$
**for** t = 0:maxIter **do**
  sample a coordinate $n$ randomly from $1 \dots N$.
  optimize $g$ w.r.t. that coordinate:

$$u^\star \leftarrow \arg\min{}^{1}_{u\in\mathbb{R}}\, g\big(\alpha_1^{(t)}, \dots, \alpha_{n-1}^{(t)}, u, \alpha_{n+1}^{(t)}, \dots, \alpha_N^{(t)}\big)$$

  update
$$\alpha_n^{(t+1)} \leftarrow u^\star$$
$$\alpha_{n'}^{(t+1)} \leftarrow \alpha_{n'}^{(t)} \text{ for } n' \neq n \quad \textit{(unchanged)}$$
**end for**

---

[1] The pseudocode here is for coordinate **de**scent, that is to minimize a function. For the equivalent problem of maximizing (coordinate **a**scent), either change this line to arg max, or use the arg min of minus the objective function.

## Issues with SVM

- There is no obvious probabilistic interpretation of SVM.
- Extension to multi-class is non-trivial
  (see Section 14.5.2.4 of KPM book).