Annotated Version

Machine Learning Course - CS-433

Support Vector Machines

Nov 1, 2016

changes by Martin Jaggi 2016

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

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

Support Vector Machine

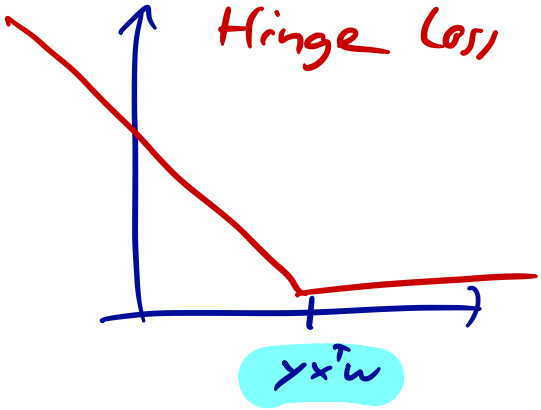
Throughout, we will work with a classification problem and assume[a] that the labels $y_n \in \{\pm 1\}$.
(This is in contrast to logistic regression, where we have used the convention $y_n \in \{0, 1\}$.)
We again write $\mathbf{x}_n$ for datapoint n, and assume that all constructed features and a potential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

LR: $y \in \{0, 1\}$
SVM: $y \in \{-1, 1\}$

The SVM optimizes the following cost:

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

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



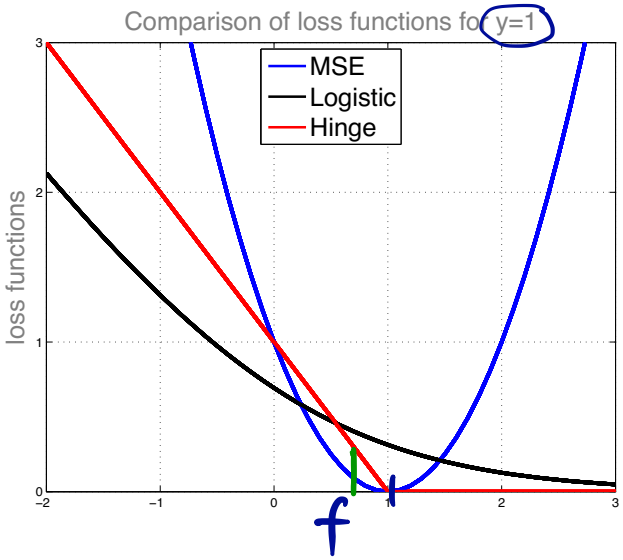
[a] Note that for any use-case, the labels can be converted accordingly before training, and after prediction.

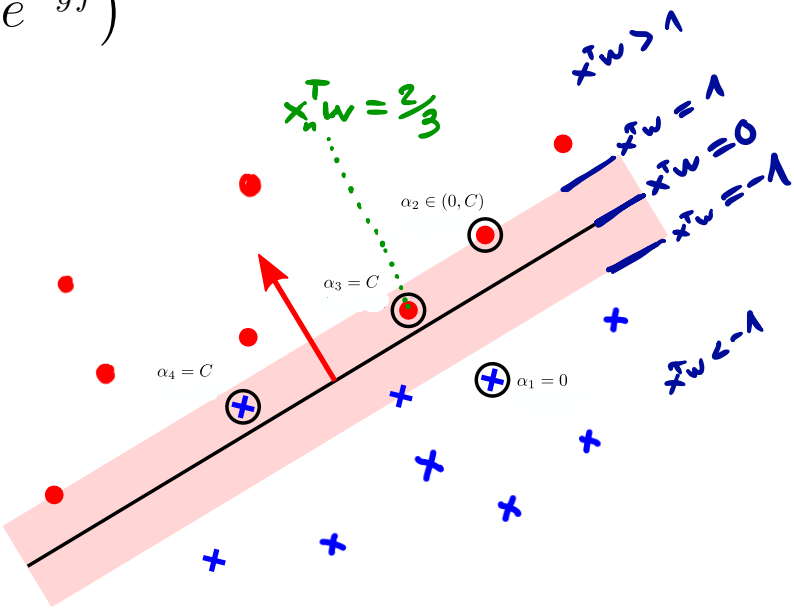
Hinge vs MSE vs Logistic

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

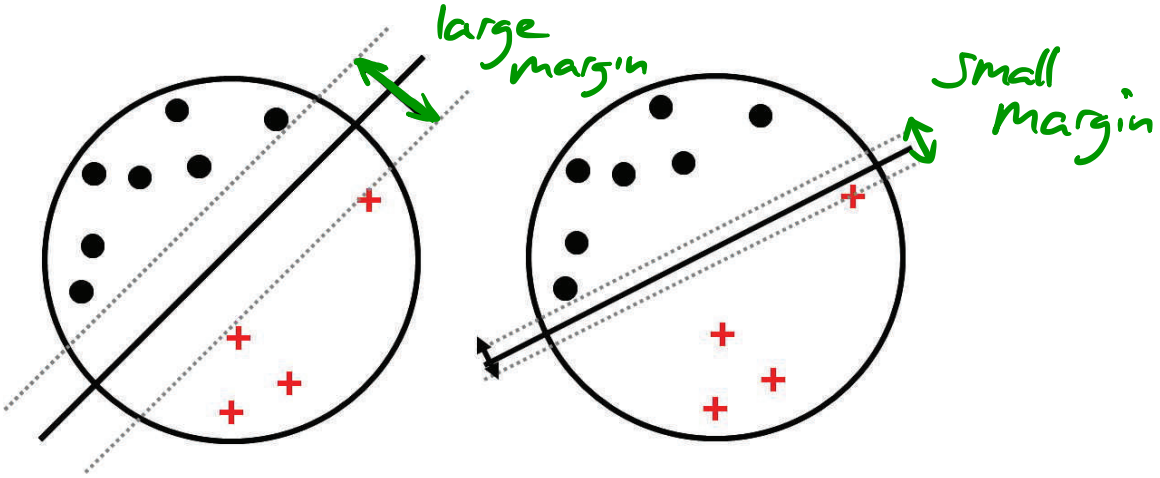
$f = x_n^T w$

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

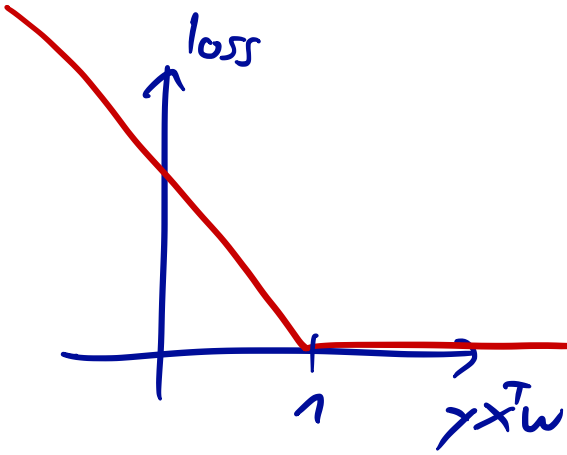




Notice the margin in the Hinge loss.
SVM is a maximum margin method.



Optimization



Is this function convex? Is it differentiable? (in $\mathbf{w}$)

$$\min_{\mathbf{w}} \sum_{n=1}^{N} \left[1 - y_n \mathbf{x}_n^\top \mathbf{w}\right]_+ + \tfrac{\lambda}{2}\|\mathbf{w}\|^2 \qquad \mathcal{L}(w)$$

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

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

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 $G(\mathbf{w}, \boldsymbol{\alpha})$?

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

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

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

equal $\forall n$

$0 \le \alpha_n \le 1$

So we can rewrite the SVM problem as:

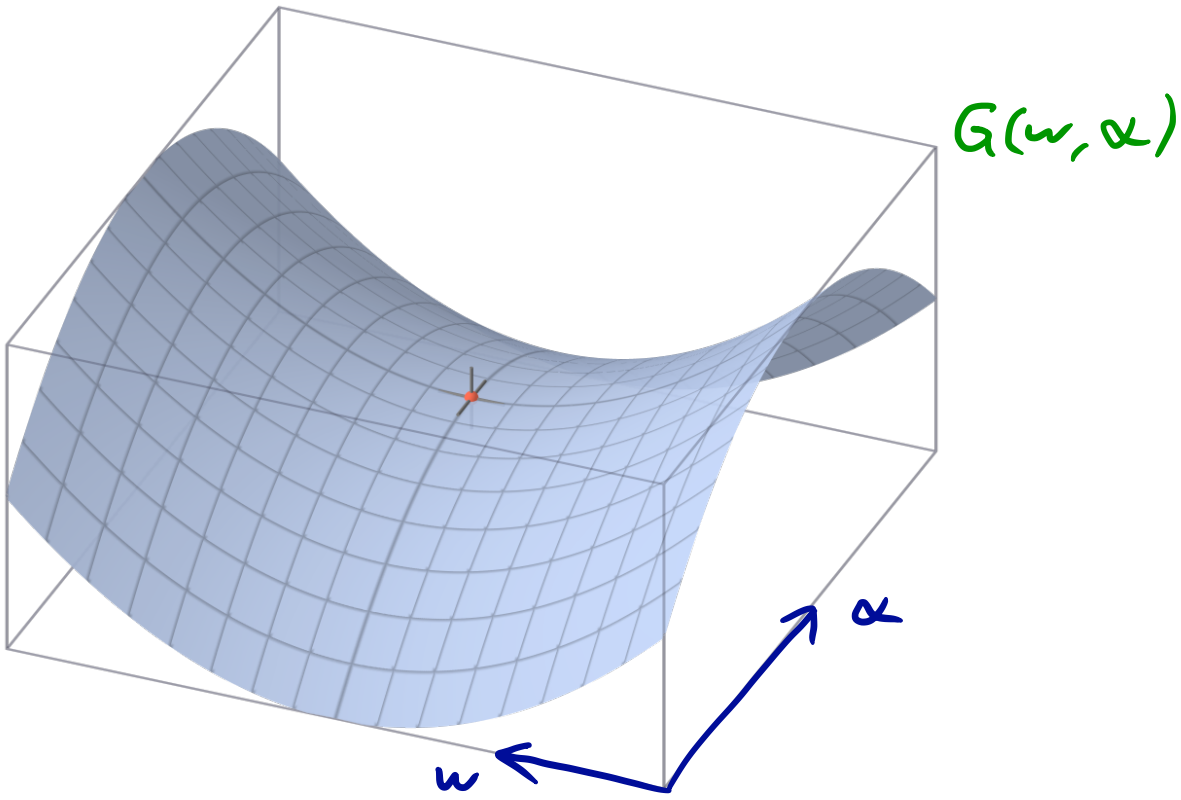
$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha} \in [0,1]^N} \underbrace{\sum_{n=1}^{N} \alpha_n (1 - y_n \mathbf{x}_n^\top \mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2}_{=: G(\boldsymbol{w}, \boldsymbol{\alpha})}$$

$\mathcal{L}(w)$

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

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.

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



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

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

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

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

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

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

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

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

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

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

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 $\mathbf{Q} := \text{diag}(\mathbf{y})\mathbf{X}^\top\mathbf{X}\text{diag}(\mathbf{y})$. Optimization is super easy using coordinate descent (or ascent) (changing one α_n variable a time).

optimize only one α_n at a time.

α_n

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

$K_{ij} = x_i^\top x_j$

$\phi(x_i)^\top \phi(x_j)$

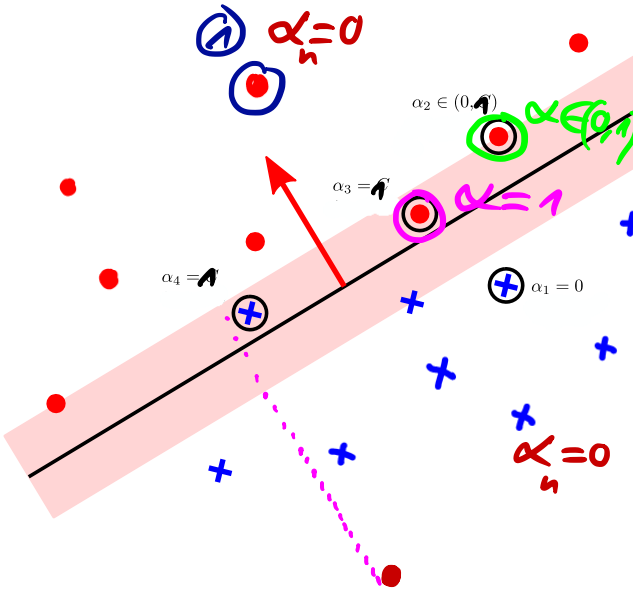
(3) The solution $\boldsymbol{\alpha}$ is typically sparse, and is non-zero only for the training examples that are instrumental in determining the decision boundary.

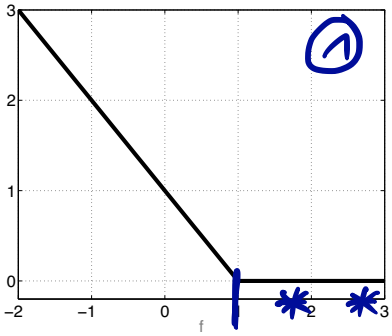
Recall that α_n is the slope of lines that are lower bounds to the Hinge loss.

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

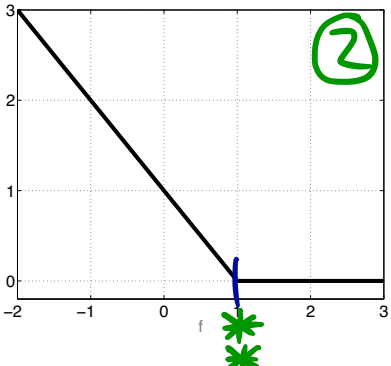
There are 3 kinds of data vectors $\mathbf{x}_n$.

1. Non-support vectors. Examples that lie on the correct side outside the margin, so $\alpha_n = 0$.
2. Essential support vectors. Examples that lie just on the margin, therefore $\alpha_n \in (0, 1)$.
3. Bound support vectors. Examples that lie strictly inside the margin, or on the wrong side, therefore $\alpha_n = 1$.

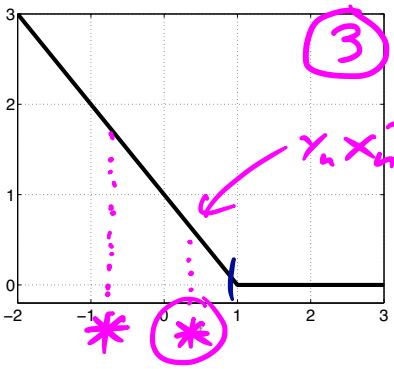




(c) Non-SV



(d) Essential SV

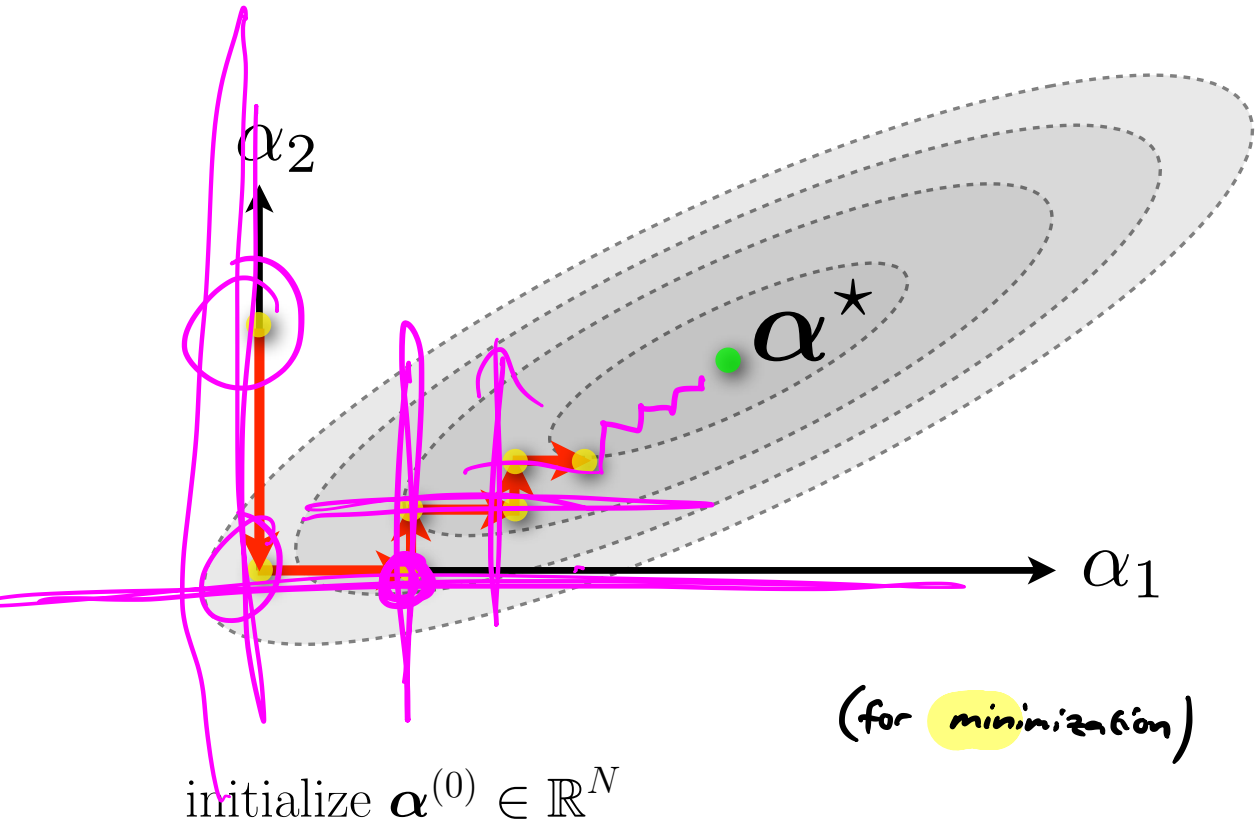


(e) Bound SV

Coordinate Descent

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

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



(for minimization)

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:

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

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

(for maximization, use argmax instead)

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).

ToDo

1. Understand and visualize hinge loss and the margin.
2. Get comfortable with duality. Work out the derivation for SVM.
3. Implement SGD and coordinate descent (on the dual) for SVM.
4. What does "support vector" mean? Why do they arise? Where do they lie in the data space?
5. Read about SVM for regression (section 14.5.1 of KPM).
6. Read Section 14.5.2.4 of KPM book and understand why extension of SVM to multi-class is non-trivial.
7. Read about maximum-margin methods in section 14.5.2.2 of KPM book.
8. Resource: SVM tutorial by Christopher J.C. Burges at research.microsoft.com/pubs/67119/svmtutorial.pdf