Machine Learning Course - CS-433

Support Vector Machines

November 6th, 2018

changes by Martin Jaggi 2016

minor changes by Martin Jaggi 2017

changes by Ruediger Urbanke 2018

Last updated: November 5, 2018

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

Let us re-consider binary classification with data pairs $(\mathbf{x}_n, y_n)$, $y_n \in \{0, 1\}$. As we had discussed, if we used least squares (not recommended!) and ignore right now any potential regularization term this would lead us to the minimization

$$\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2.$$

If instead we used logistic regression and then optimized the log-likelihood, we would solve

$$\min_{\mathbf{w}} \sum_{n=1}^{N} \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) - y_n \mathbf{x}_n^\top \mathbf{w}.$$

In the following it will be slightly more convenient to assume that $y_n \in \{\pm 1\}$, where we have the mappings $0 \leftrightarrow 1$ and $1 \leftrightarrow -1$. If we rewrite the above two minimization problems after this transform we get

$$\min_{\mathbf{w}} \sum_{n=1}^{N} (1 - y_n \mathbf{x}_n^\top \mathbf{w})^2 = \min_{\mathbf{w}} \sum_{n=1}^{N} \mathrm{MSE}(\mathbf{x}_n^\top \mathbf{w}, y_n),$$

$$\min_{\mathbf{w}} \sum_{n=1}^{N} \log(1 + e^{-y_n \mathbf{x}_n^\top \mathbf{w}}) = \min_{\mathbf{w}} \sum_{n=1}^{N} LogisticLoss(\mathbf{x}_n^\top \mathbf{w}, y_n),$$

where

$$\mathrm{MSE}(z, y) = (1 - yz)^2,$$
$$\mathrm{LogisticLoss}(z, y) = \log(1 + \exp(-yz)).$$

Here, z is the prediction based on the given data point and y is the label associated to the the data point. We get support vector machines (SVMs), if instead we use the losss

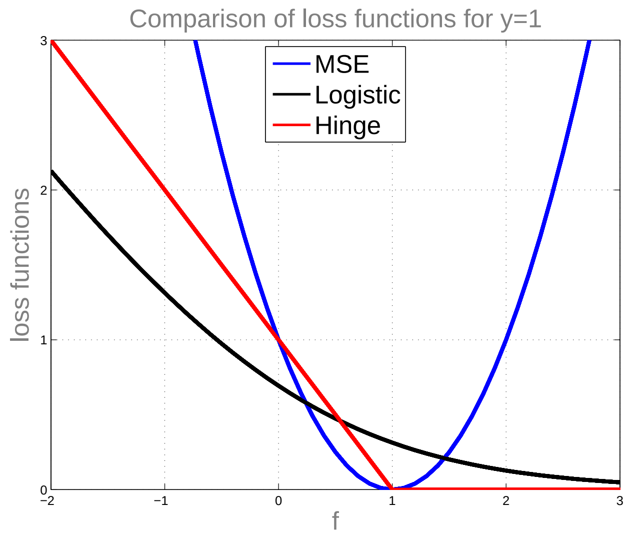
$$\text{Hinge}(z, y) = [1 - yz]_+ = \max\{0, 1 - z\}.$$

Support Vector Machine

In the sequel we will assume that the labels y_n takes values in $\{\pm 1\}$. SVMs correspond to the following optimization:

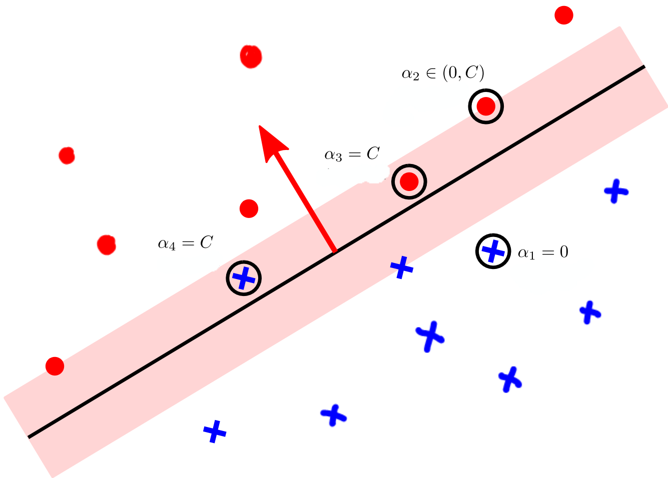
$$\min_{\mathbf{w}} \sum_{n=1}^{N} \left[1 - y_n \mathbf{x}_n^\top \mathbf{w}\right]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

The next figure compares the three cost functions: Note that

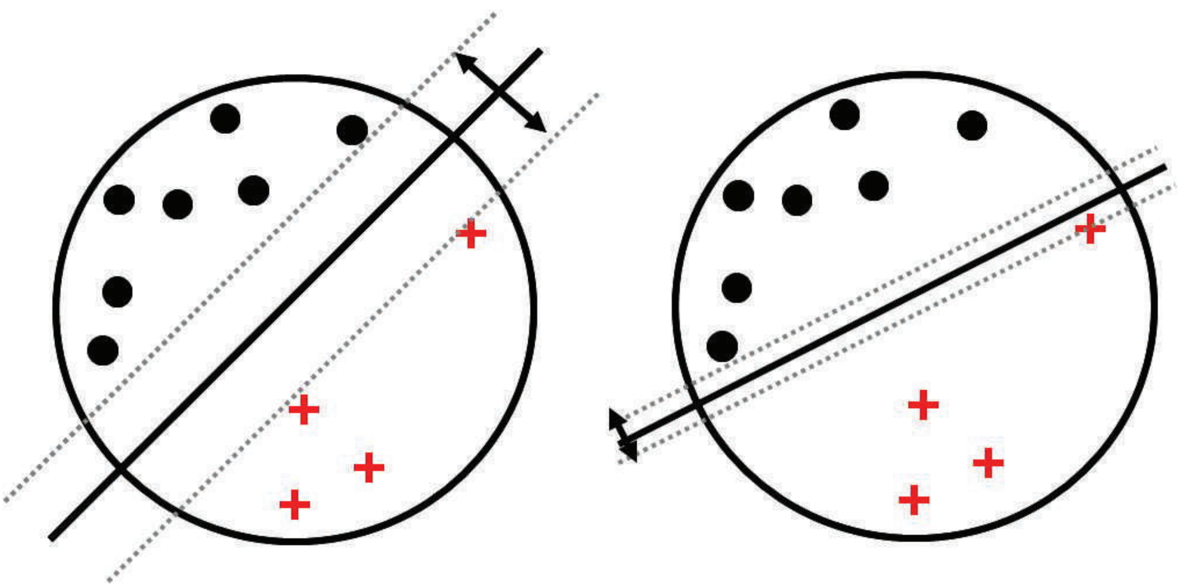


for least squares we incur a cost whenever we fail to represent the desired value exactly and the cost is symmetric around the target value. For logistic regression we always incur a cost but the cost is asymmetric – it becomes smaller the further we are "on the right side" and it becomes larger the further we are "on the wrong side." The hinge loss acts differently.

Once we are “sufficiently far” on the right side we no longer pay a cost. But if we are not yet far enough on the correct side or if we are on the wrong side we do pay a cost and the cost increases linearly the worse we are. The region on the “correct side” where we still pay a cost is called the *margin*. This margin is shown explicitly in the following figure.



To get a better insight of what this algorithm does, consider the case of separable data. It is then clear that the optimization will pick that hyperplane that maximizes the margin.



Optimization

Now where we have established *what function* we are optimizing, let us look at the question *how* we can optimize it efficiently.
Note that the function is convex and and has a subgradient (in $\mathbf{w}$).

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

We can thefore use SGD! (with subgradients).

Duality: The big picture

We have just seen that we can use SGD in order to find the optimal parameters for the SVM. We will now discuss an alternative but equivalent formulation via the concept of *duality*. In some cases this leads to a more efficient implementation. But perhaps more importantly, once we have derived this alternative representation it will point us naturally to a more general formulation. This is called the *kernel trick*. We will explicitly discuss this technique in a separate lecture.
Let us say that we are interested in minimizing a function $\mathcal{L}(\mathbf{w})$. Assume that we can define an auxiliary function $G(\mathbf{w}, \boldsymbol{\alpha})$ so that

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

We can therefore solve our original problem by solving

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

We call this the *primal* problem. In some cases it might be much easier to find

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

We call this the *dual* problem. This leads us naturally to the following questions:

1. How do we find a suitable $G(\mathbf{w}, \boldsymbol{\alpha})$?
2. When is it OK to switch $\min\limits_{\boldsymbol{w}}$ and $\max\limits_{\boldsymbol{\alpha}}$?
3. When is the dual easier to optimize than the primal?

Q1: How do we find a suitable $G(\mathbf{w}, \boldsymbol{\alpha})$? There is a general theory on this topic (see e.g., Bertsekas' "Nonlinear Programming" for more formal details). But rather than talk about this in the abstract, let us look at our specific problem. We have

$$[z]_+ = \max\{0, z\} = \max_{\alpha \in [0,1]} \alpha z.$$

Therefore,

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

So we can rewrite the SVM problem as:

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

Note that $G(\boldsymbol{w}, \boldsymbol{\alpha})$ is convex in $\mathbf{w}$ and linear, hence concave, in $\boldsymbol{\alpha}$.

Q2: When is it OK to switch max and min?

Note that it is always true that

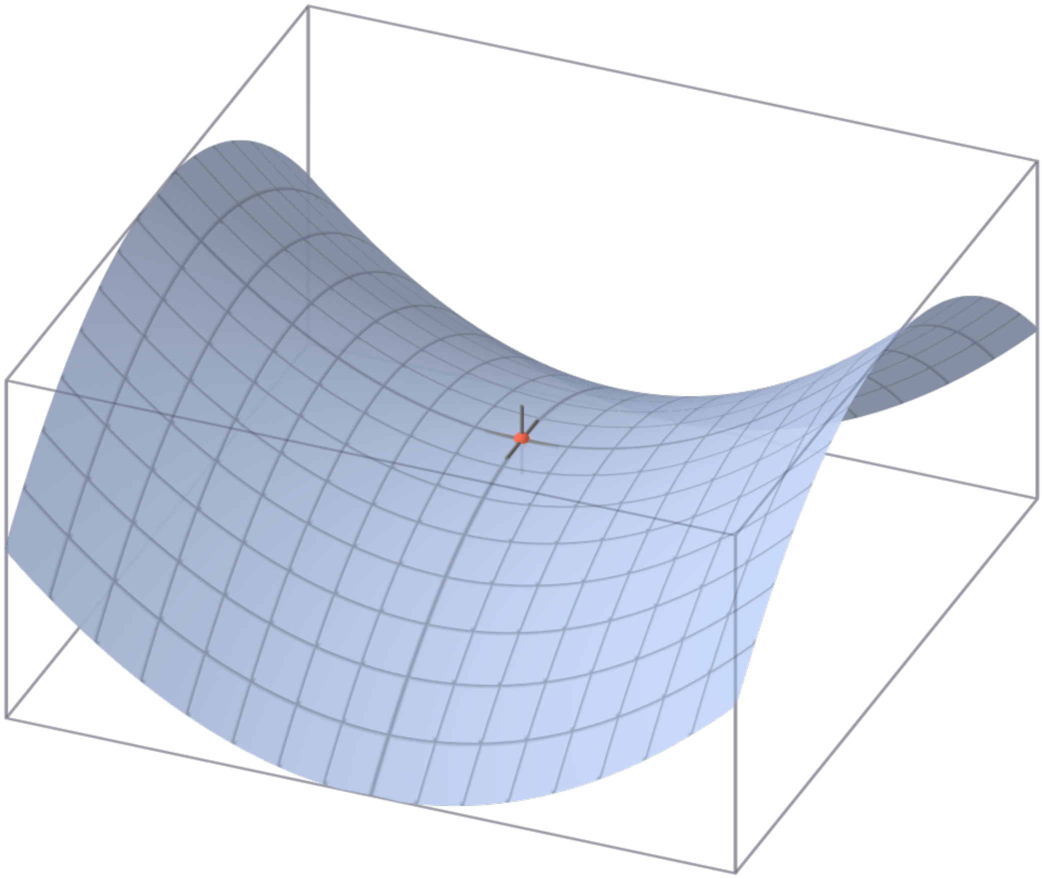
$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}} G(\mathbf{w}, \boldsymbol{\alpha}) \leq \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}} G(\mathbf{w}, \boldsymbol{\alpha}).$$

This is easy to see:

$$\begin{aligned}
\min_{\boldsymbol{w}'} G(\mathbf{w}', \boldsymbol{\alpha}) &\leq G(\mathbf{w}, \boldsymbol{\alpha}) \ \ \forall \mathbf{w}, \boldsymbol{\alpha}, \Leftrightarrow \\
\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}'} G(\mathbf{w}', \boldsymbol{\alpha}) &\leq \max_{\boldsymbol{\alpha}} G(\mathbf{w}, \boldsymbol{\alpha}) \ \ \forall \mathbf{w} \Leftrightarrow \\
\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}'} G(\mathbf{w}', \boldsymbol{\alpha}) &\leq \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}} G(\mathbf{w}, \boldsymbol{\alpha}).
\end{aligned}$$

We get equality if $G(\mathbf{w}, \boldsymbol{\alpha})$ is a continuous function that is convex in $\mathbf{w}$, concave in $\boldsymbol{\alpha}$, and the domain of $\mathbf{w}$ and $\boldsymbol{\alpha}$ are both compact and convex. I.e., in this case we have

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



In other words, we get equality if we have functions that look like saddles as in the previous figure.
For SVMs the condition is fulfilled and we can swith the min and max. This leads to the following formulation

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

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

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

Equating this to $\mathbf{0}$, we can explicitly solve for $\mathbf{w}$ for any given $\boldsymbol{\alpha}$. We get

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

where $\mathbf{Y} := \text{diag}(\mathbf{y})$.
Plugging this $\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$ back into the saddle-point formulation (1), gives rise to the following dual optimization problem:

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

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

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

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

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

(2) Note that in the dual formulation the data only enters in the form $\mathbf{K} := \mathbf{X}\mathbf{X}^\top$. We say the this formulation is *kernelized*. As we will discuss in the next lecture, this has a very pleasing consequence.

(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 the function of the parameters α_n:

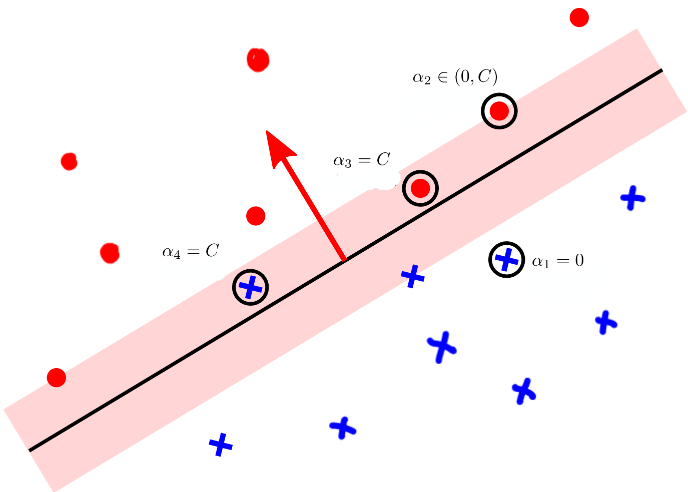
$$[1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ = \max_{\alpha_n \in [0,1]} \alpha_n (1 - y_n \mathbf{x}_n^\top \mathbf{w})$$

From this formulation we can see that there are three distinct cases we should consider:

a) Examples that lie on the correct side outside the margin, so $\alpha_n = 0$. We call those $\mathbf{x}_n$ *non-support* vectors.

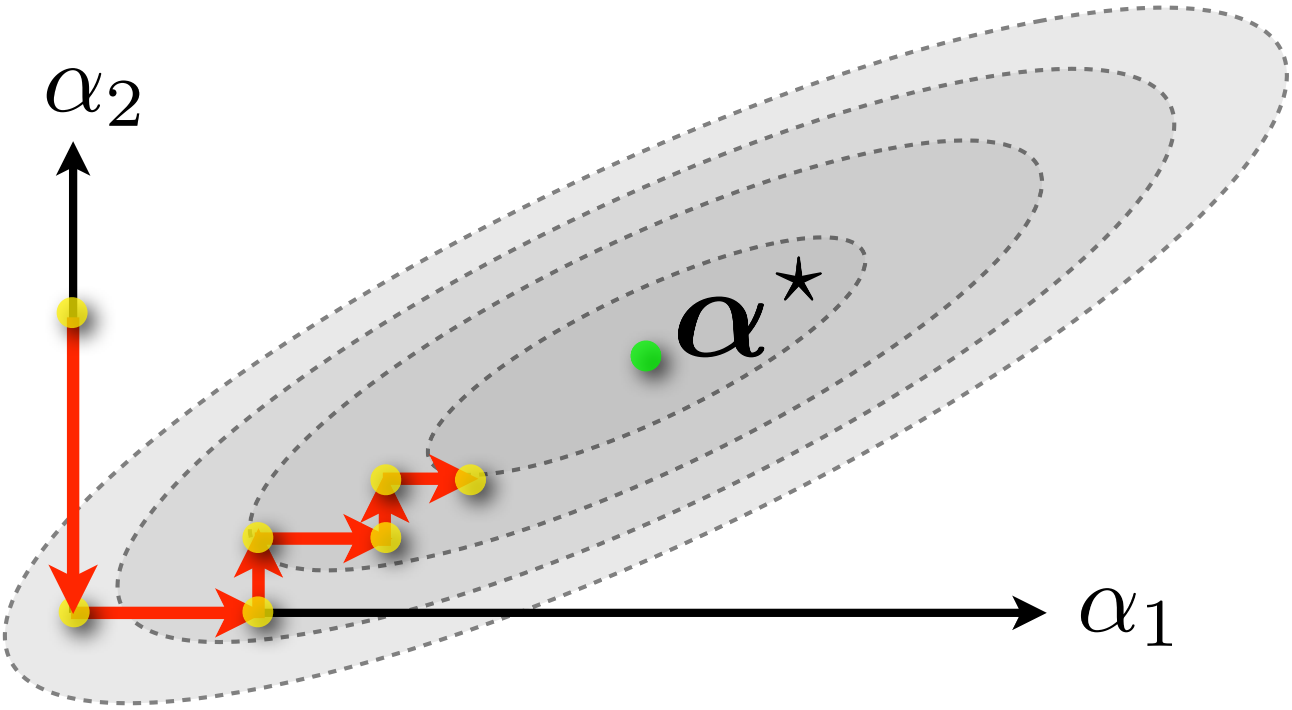
b) Examples that lie on the correct side and just "on the margin". I.e., for those examples we have $y_n \mathbf{x}_n^\top \mathbf{w} = 1$. Therefore $\alpha_n \in (0, 1)$. We call those $\mathbf{x}_n$ *essential support vectors.*

c) Examples that lie strictly inside the margin, or on the wrong side, therefore $\alpha_n = 1$. We call those $\mathbf{x}_n$ *bound support vectors.*



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 $1 \ldots N$.
  optimize g w.r.t. that coordinate:

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

  update $\alpha_n^{(t+1)} \leftarrow u^\star$

[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.

$$\alpha_{n'}^{(t+1)} \leftarrow \alpha_{n'}^{(t)} \text{ for } n' \neq n \quad \textit{(unchanged)}$$

end for

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).