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 7, 2018

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

Let us re-consider binary classification with data pairs $(\mathbf{x}_n, \tilde{y}_n)$, $\tilde{y}_n \in \{0, 1\}$. We use here the notation $\tilde{y}_n$ to denote variables that take values in $\{0, 1\}$ since soon we will transform our labels to take values in $\{\pm 1\}$ and we want to avoid confusion. As we had discussed, if we use least squares (not recommended!) and ignore right now any potential regularization term this lead us to the minimization

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

If instead we use logistic regression and optimize the log-likelihood, we solve

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

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 $1 \leftrightarrow 1$. We can write this compactly as $y_n = 2\tilde{y}_n - 1$, or equivalently, $\tilde{y}_n = \frac{1}{2}(y_n + 1)$.

Consider first the least squares problem. Assume, as always, that the feature vector contains the constant feature 1 and that this is component 0. Define $\tilde{\mathbf{w}} = \frac{1}{2}(\mathbf{w} + e_0)$, where e_0 is the vector of lengnth D (the dimension of the feature vector) that has a 1 at component 0 and 0 at all other components. Note that this defines a one-to-one mapping between $\mathbf{w}$ and

$\tilde{\mathbf{w}}$. We then have

$$\begin{aligned}
4\sum_{n=1}^{N}(\tilde{y}_n - \mathbf{x}_n^\top \tilde{\mathbf{w}})^2 &= 4\sum_{n=1}^{N}(\frac{1}{2}(y_n+1) - \frac{1}{2}\mathbf{x}_n^\top(\mathbf{w}+e_0))^2 \\
&= \sum_{n=1}^{N}((y_n+1) - \mathbf{x}_n^\top(\mathbf{w}+e_0))^2 \\
&= \sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \\
&\overset{(a)}{=} \sum_{n=1}^{N}(1 - y_n\mathbf{x}_n^\top \mathbf{w})^2 \\
&= \sum_{n=1}^{N}\mathrm{MSE}(\mathbf{x}_n^\top \mathbf{w}, y_n),
\end{aligned}$$

where

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

In step (a) we have used the fact that $y_n\{\pm 1\}$ so that $y_n^2 = 1$. In a similar manner, for logistic regression we have

$$\begin{aligned}
\sum_{n=1}^{N}\log(1+e^{\mathbf{x}_n^\top \mathbf{w}}) - \tilde{y}_n\mathbf{x}_n^\top \mathbf{w} &= \sum_{n=1}^{N}\log(1+e^{-y_n\mathbf{x}_n^\top \mathbf{w}}) \\
&= \sum_{n=1}^{N}\mathrm{LogisticLoss}(\mathbf{x}_n^\top \mathbf{w}, y_n),
\end{aligned}$$

where

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

This is most easily seen by checking the two cases $\tilde{y}_n = 0 \leftrightarrow y_n = -1$ and $\tilde{y}_n = 1 \leftrightarrow y_n = 1$ separately.
Note that above, z is the prediction based on the given data point and y is the label associated to the data point. We get support vector machines (SVMs), if instead we use the losss

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

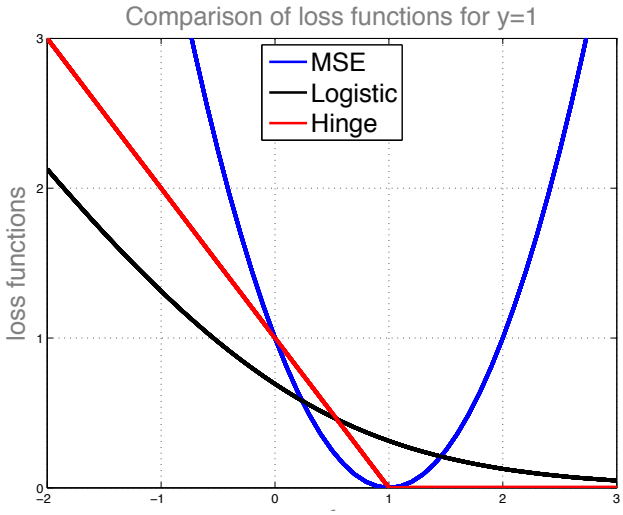
and add a regularization term.

Support Vector Machine

As just mentioned, SVMs correspond to the following optimization problem:

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

The next figure compares the cost functions of the mean-squared loss, the logistic loss, and the hinge loss. Note that

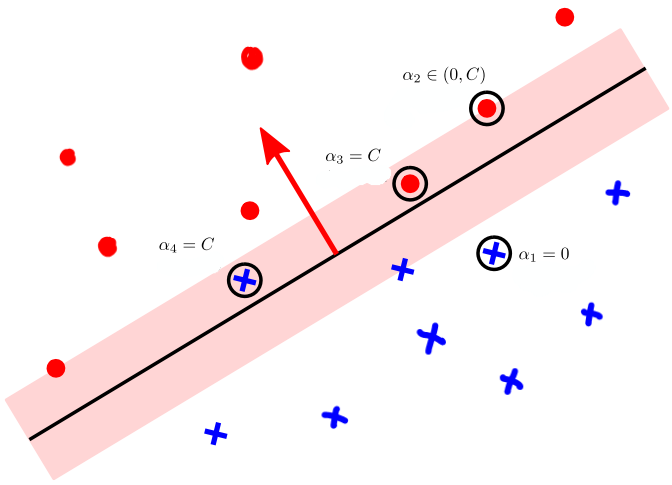


for least squares we incur a cost whenever we fail to represent the desired value exactly and the cost is symmetric around

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 further we are away.

In the figure below you see a region in pink. This region is called the "margin." It is defined as follows: Take the normal vector $\mathbf{w}$ that defines the hyperplane. Look at all feature vectors $\mathbf{x}$ so that $|\mathbf{x}^\top \mathbf{w}| \leq 1$. This is the margin. Note that the margin does not only depend on the direction of the vector $\mathbf{w}$ but also its norm. In fact the total width of the margin is equal to $2/\|\mathbf{w}\|$.

If you are looking for an interpretation: It is the region where our prediction is not yet sufficiently "confident" (assuming that we get the sign right). The intuition is strongest if we

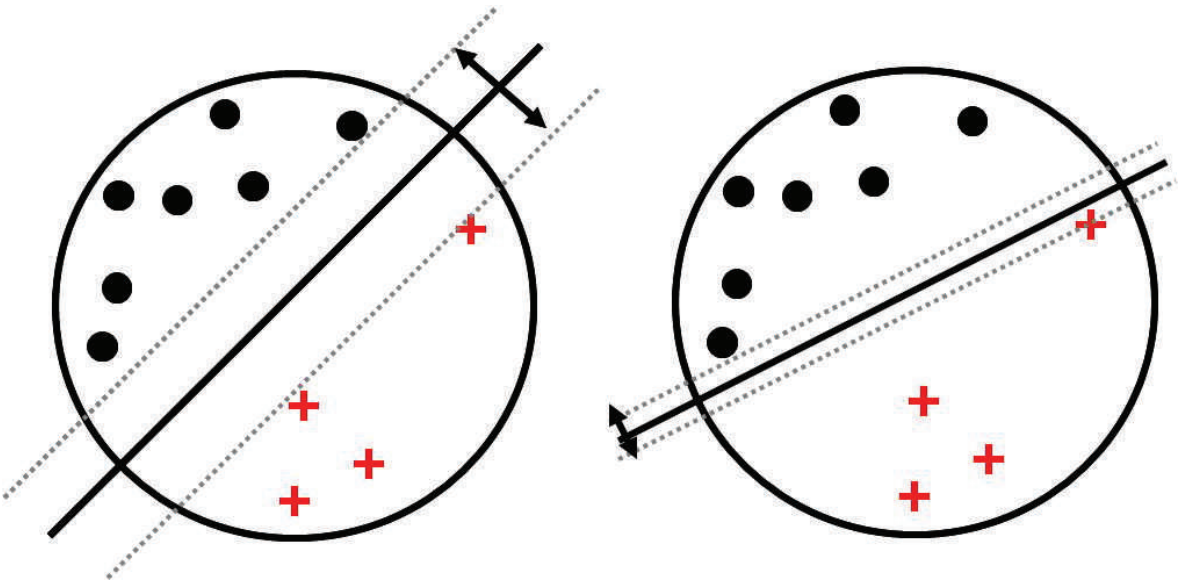


look at the case of separable data. This is shown in the following two figures. Assume that λ is small so that the cost function is dominated by the sum over the hinge losses

on the left. What will then the optimal $\mathbf{w}$ look like? It is then clear that we want the following:

1. A separating hyperplane.
2. A scaling of $\mathbf{w}$ so that no point of the data is in the margin.
3. That separating hyperplane and scaling for which the margin is the largest.

Conditions (1) and (2) ensure that there is no cost incurred in the first expression (the sum over the terms $\left[1 - y_n \mathbf{x}_n^\top \mathbf{w}\right]_+$). Since by assumption that λ is small this is the dominant term, we cannot hope to do better than having this term to be 0. The condition (3) ensures that we have the minimum possible cost associated for the regularization term, i.e. the minimal possible normed squared $\|\mathbf{w}\|^2$. Geometrically this corresponds to a hyperplane with maximal "spacing" to the left and right, i.e., a hyperplane with maximal 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 $\mathbf{w}$):

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

We can thefore use SGD with subgradients. This is good news!

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 $\min_{\boldsymbol{w}}$ and $\max_{\boldsymbol{\alpha}}$?
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:

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

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

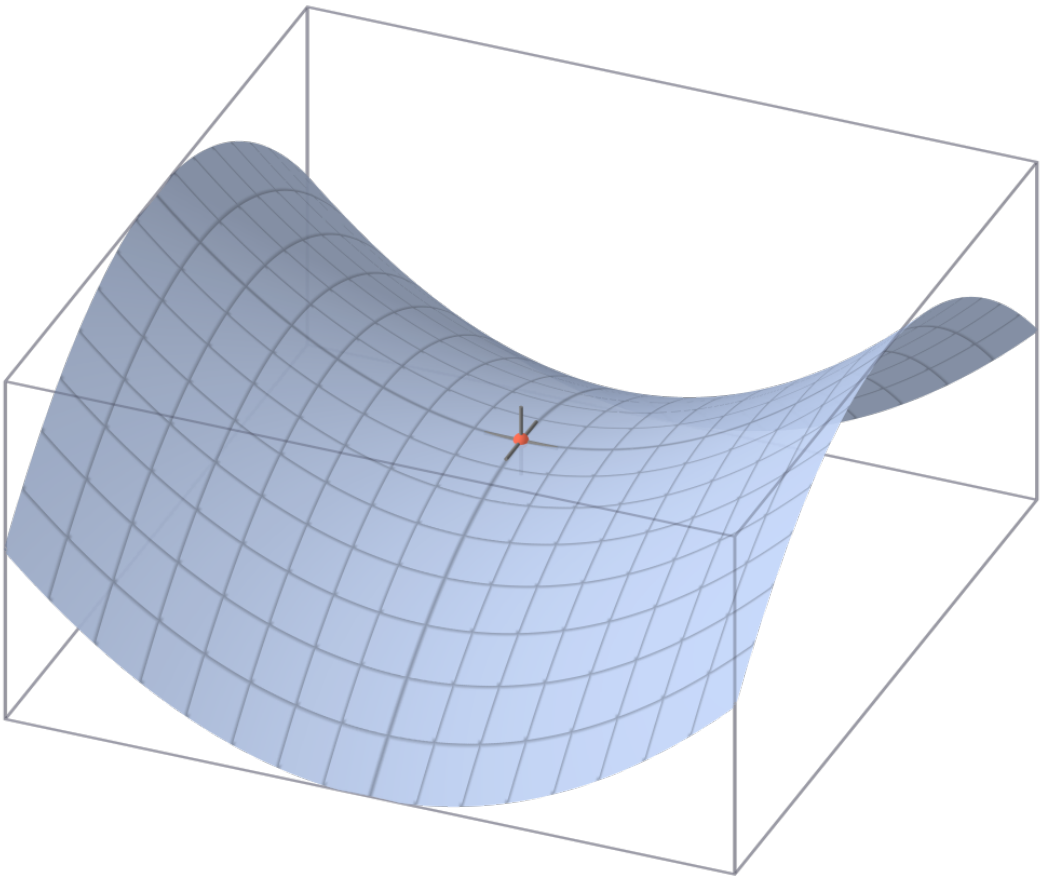
$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}} G(\mathbf{w}, \boldsymbol{\alpha}) \leq \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}} G(\mathbf{w}, \boldsymbol{\alpha}).$$

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 switch 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 $\mathbf{K} := \mathbf{X}\mathbf{X}^\top$. This product $\mathbf{X}\mathbf{X}^\top$ is called the "kernel." We hence often say that 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 how the parameters α_n were introduced:

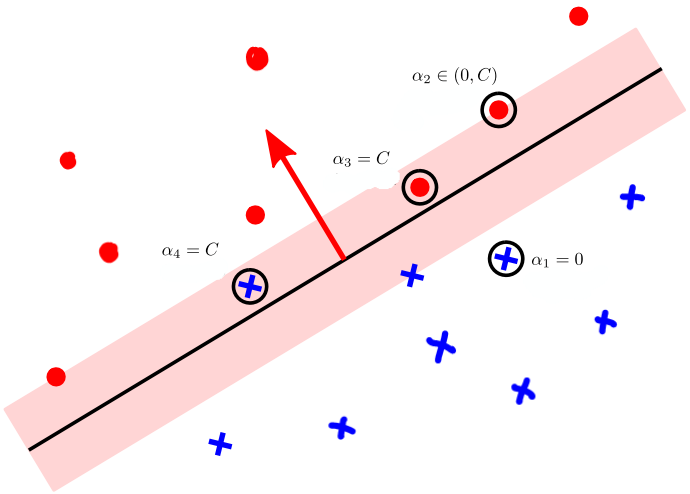
$$[1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ = \max_{\alpha_n \in [0,1]} \alpha_n (1 - y_n \mathbf{x}_n^\top \mathbf{w})$$

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

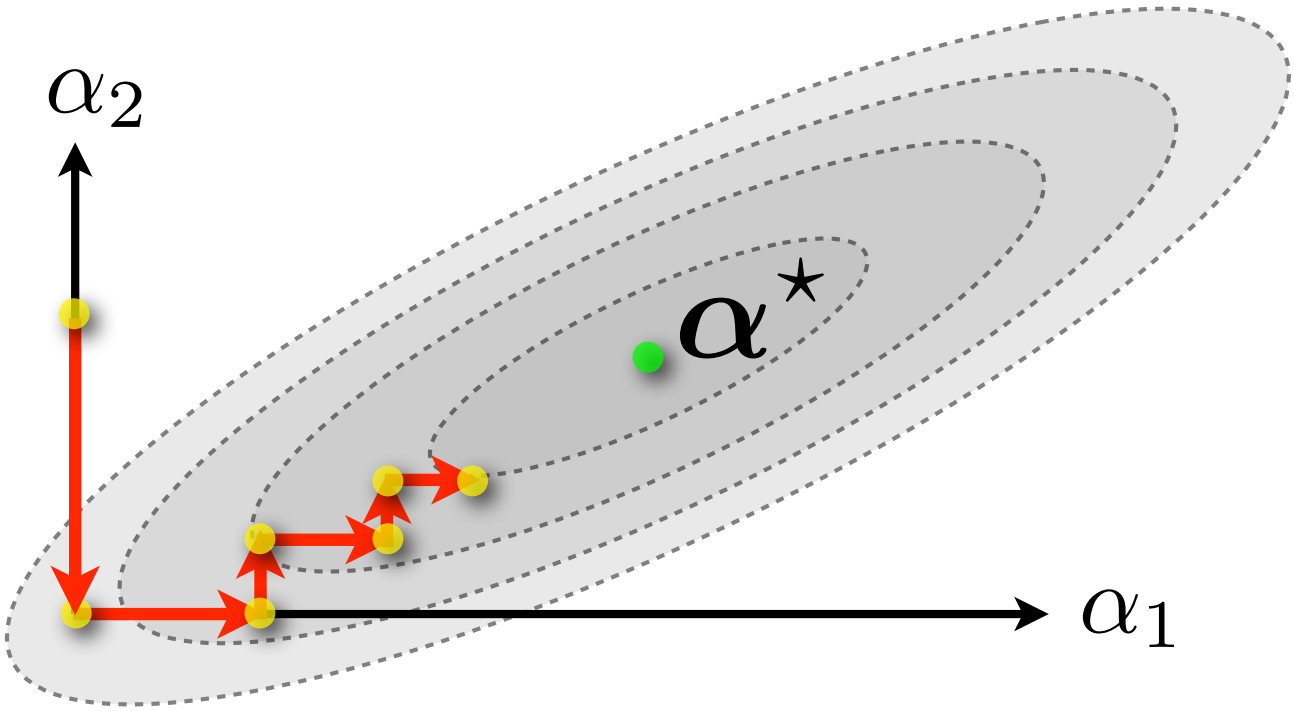
a) Examples that lie on the correct side and outside the margin. For those $1 - y_n \mathbf{x}_n^\top \mathbf{w} < 0$. Hence, $\alpha_n = 0$. We call those $\mathbf{x}_n$ *non-support* vectors.

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



The above consideration explains why we expect most α_n to be zero.



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

[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

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

$\alpha_{n'}^{(t+1)} \leftarrow \alpha_{n'}^{(t)}$ for $n' \neq n$ *(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).

---

this line to arg max, or use the arg min of minus the objective function.