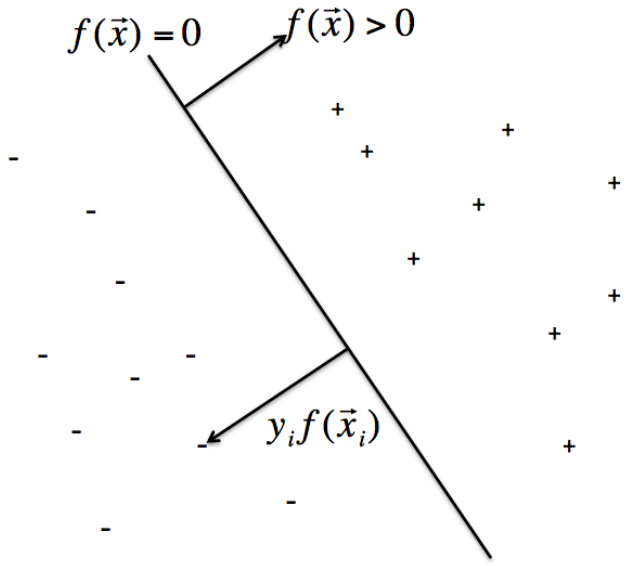
Support Vector Machines

Duke Course Notes
Cynthia Rudin

Let's start with some intuition about margins.

The margin of an observation $\mathbf{x}_i$ = "distance" from observation to decision boundary
$= y_i f(\mathbf{x}_i)$



The margin is positive if the observation is on the correct side of the decision boundary, otherwise it's negative.

Here's the intuition for SVM's:

- We want all observations to have large margins, want them to be as far from decision boundary as possible.
- That way, the decision boundary is more "stable," we are confident in all decisions.

Most other algorithms (logistic regression, decision trees, perceptron) don't generally produce large margins. (AdaBoost generally produces large margins.)

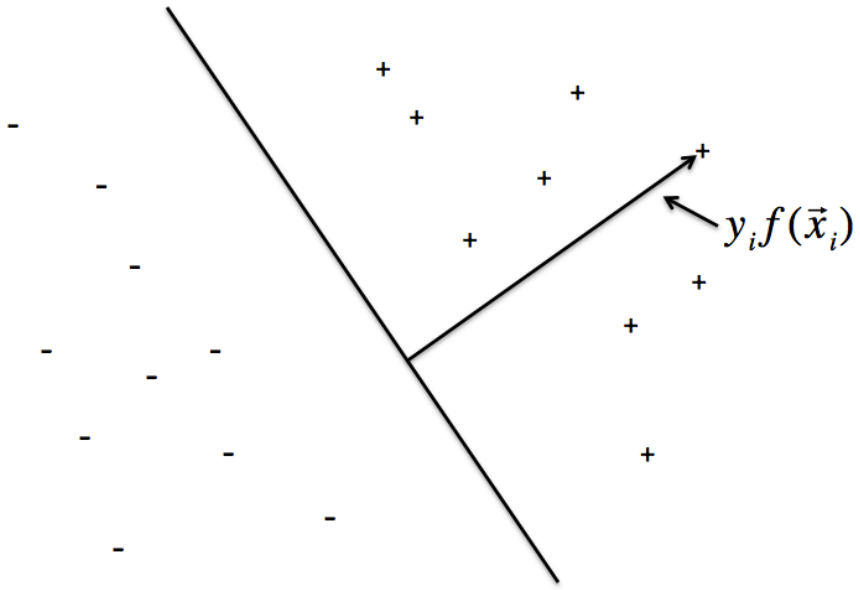
SVM's maximize the distance from the decision boundary to the nearest training observation – they maximize the minimum margin.

As in logistic regression and AdaBoost, function f is linear,

$$f(\mathbf{x}) = \sum_{j=1}^{p} \lambda_j x_{\cdot j} + \lambda_0.$$

Note that the intercept term can get swept into $\mathbf{x}$ by adding a 1 as the last component of each $\mathbf{x}$. Then $f(\mathbf{x})$ would be just $\boldsymbol{\lambda}^T\mathbf{x}$ but for this lecture we'll keep the intercept term separately because SVM handles that term differently than if you put the intercept as a separate feature. We classify $\mathbf{x}$ using $\text{sign}(f(\mathbf{x}))$.

If $\mathbf{x}_i$ has a large margin, we are confident that we classified it correctly. So we're essentially suggesting to use the margin $y_i f(\mathbf{x}_i)$ to measure the confidence in our prediction.

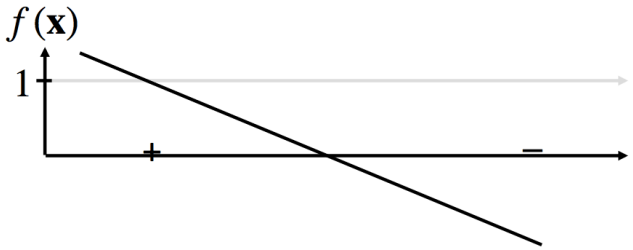


But there is a problem with using $y_i f(\mathbf{x}_i)$ to measure confidence in prediction. There is some arbitrariness about it.

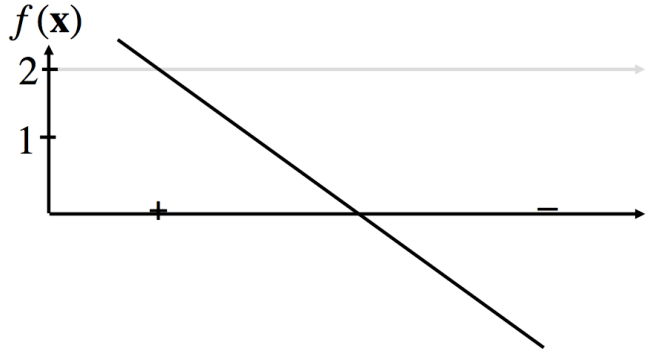
If we multiply f by 2, the decision boundary doesn't change but we become twice as confident! We can make $f(\mathbf{x})$ arbitrarily large if we can scale $\boldsymbol{\lambda}$ and λ_0

arbitrarily. So we force f to have norm 1 so that doesn't happen.

I'm going to tip the previous picture a different way. The axes in the above picture represent features (maybe the first component of $\mathbf{x}$ and the second). There's a third important direction, which is out of the page, $f(\mathbf{x})$, but you can only see marked the decision boundary where $f(\mathbf{x}) = 0$. Remember that f is positive where the positive points are and negative where the negative points are. In the pictures below, the horizontal axis is along feature space and the vertical axis is function values $f(\mathbf{x})$. In the first picture you can see that the functional margin $y_i f(\mathbf{x}_i)$ for the positive example on the left is 1, which is $y_i f(\mathbf{x}_i)$.



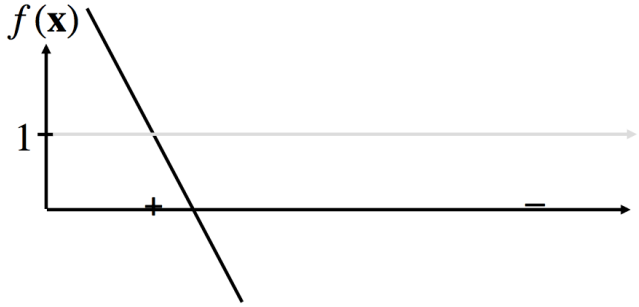
Now double $f(\mathbf{x})$, and $y_i f(\mathbf{x}_i)$ is 2, but the decision boundary didn't change. The absolute slope is higher though.



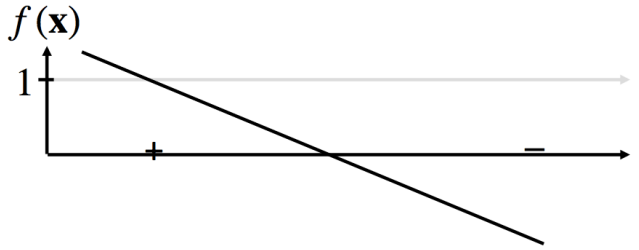
In this case $y_i f(\mathbf{x}_i)$ doesn't tell us anything about how far the point is from the decision boundary. We could fix this in a couple ways. One way is to fix the absolute slope to be 1 all the time and just measure the distance to the nearest point. Then we maximize the distance to the nearest point to maximize a geometric margin that is meaningful.

Another way is to just force the nearest point to have functional margin $y_i f(\mathbf{x}_i)$ equal to 1. What happens then? Let's say the nearest point is very close to

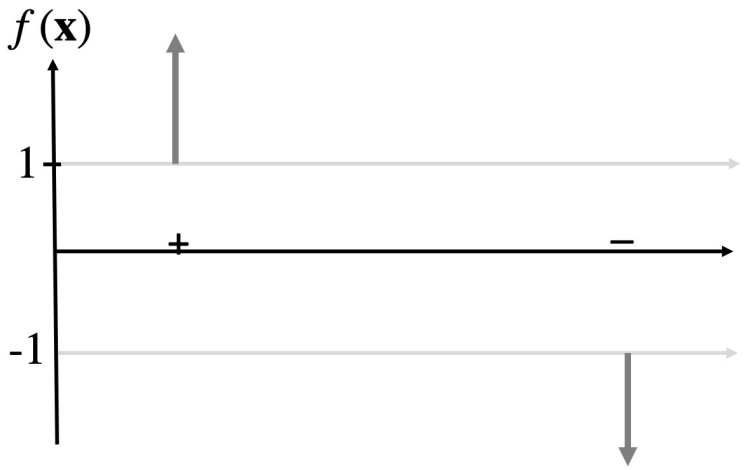
the decision boundary. Then in order to force $y_i f(\mathbf{x}_i)$ to be 1, then the absolute slope of f would need to be very large!



Now what do we tell an algorithm in order to keep the decision boundary away from the nearest training examples? You could ask the algorithm to choose f with a small absolute slope. It would then prefer the picture below to the one above.



What would the algorithm do if we told it to have all of its functional margins $y_i f(x_i) \geq 1$, and asked it to have the lowest absolute slope? The picture below shows gray arrows where the functional margins $y_i f(x_i)$ are at least 1. When tell it to minimize the slope, it ends up maximizing the distance to the nearest points.



We just showed that if you fix the absolute slope, you can maximize the geometric margin – this gives the same answer as if you restrict all the functional margins to be above one and minimize the slope.

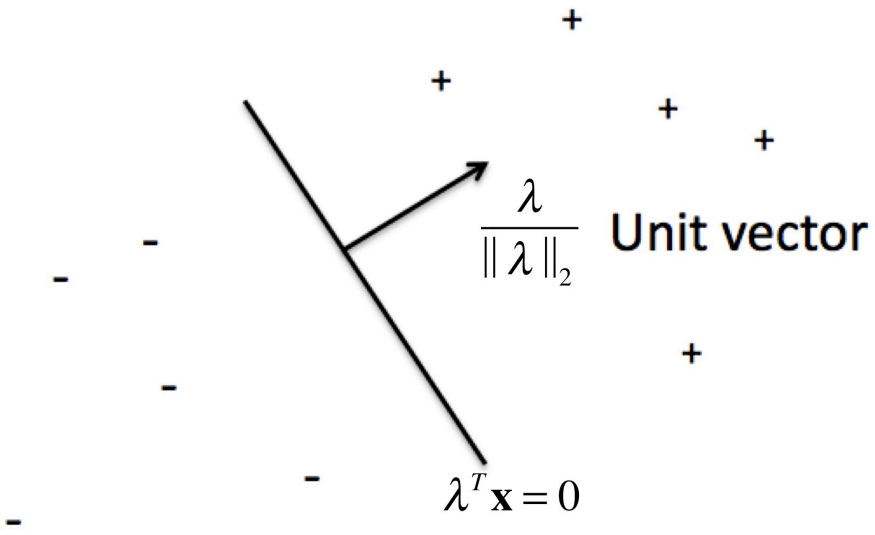
So there is a tradeoff between the margin and the slope.

I say often that the margin is the same as the distance to the decision boundary, but I never really proved that. Let's do it.

Geometric perspective: *The "functional margin" $y_i f(x_i)$ is the same as the geometric margin, which is the distance from observation i to the decision boundary.*

Let's show this.

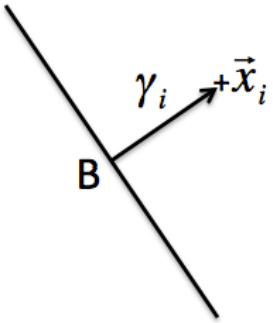
I set the intercept to zero for this picture (so the decision boundary passes through the origin):



The decision boundary are $\mathbf{x}$'s where $\boldsymbol{\lambda}^T\mathbf{x} = 0$. That means the unit vector for $\boldsymbol{\lambda}$ must be perpendicular to those $\mathbf{x}$'s that lie on the decision boundary.

Now that you have the intuition, we'll put the intercept back, and we have to translate the decision boundary, so it's really the set of $\mathbf{x}$'s where $\boldsymbol{\lambda}^T\mathbf{x} + \lambda_0 = 0$.

The margin of observation i is denoted γ_i:



$\mathbf{B}$ is the point on the decision boundary closest to the positive observation $\mathbf{x}_i$. $\mathbf{B}$ is

$$\mathbf{B} = \mathbf{x}_i - \gamma_i \frac{\boldsymbol{\lambda}}{||\boldsymbol{\lambda}||_2}$$

since we moved $-\gamma_i$ units along the unit vector to get from the observation to $\mathbf{B}$.

Since $\mathbf{B}$ lies on the decision boundary, it obeys $\boldsymbol{\lambda}^T\mathbf{x} + \lambda_0 = 0$, where $\mathbf{x}$ is $\mathbf{B}$. (I wrote the intercept there explicitly). So,

$$\boldsymbol{\lambda}^T \left(\mathbf{x}_i - \gamma_i \frac{\boldsymbol{\lambda}}{||\boldsymbol{\lambda}||_2} \right) + \lambda_0 = 0$$
$$\boldsymbol{\lambda}^T \mathbf{x}_i - \gamma_i \frac{||\boldsymbol{\lambda}||_2^2}{||\boldsymbol{\lambda}||_2} + \lambda_0 = 0$$

Simplifying,

$$\begin{aligned}
\gamma_i &= \frac{\boldsymbol{\lambda}^T \mathbf{x}_i + \lambda_0}{||\boldsymbol{\lambda}||_2} \\
&=: \tilde{f}(\mathbf{x}_i) \quad \text{(this is the normalized version of } f\text{)} \\
&= y_i \tilde{f}(\mathbf{x}_i) \text{ since } y_i = 1.
\end{aligned}$$

Note that here we normalized so we wouldn't have the arbitrariness in the meaning of the margin.

If the observation is negative, the same calculation works, with a few sign flips (we'd need to move γ_i units rather than $-\gamma_i$ units).

So the "geometric" margin from the picture is the same as the "functional" margin $y_i \tilde{f}(\mathbf{x}_i)$.

Maximize the minimum margin

Support vector machines maximize the minimum margin. They would like to have all observations being far from the decision boundary. So they'll choose f this way:

$$\max_f \max_\gamma \gamma \quad \text{s.t.} \quad y_i f(\mathbf{x}_i) \geq \gamma \quad i = 1 \dots n$$

$$\max_{\gamma, \boldsymbol{\lambda}, \lambda_0} \gamma \quad \text{s.t.} \quad y_i \frac{\boldsymbol{\lambda}^T \mathbf{x}_i + \lambda_0}{||\boldsymbol{\lambda}||_2} \geq \gamma \quad i = 1 \dots n$$

$$\max_{\gamma, \boldsymbol{\lambda}, \lambda_0} \gamma \quad \text{s.t.} \quad y_i (\boldsymbol{\lambda}^T \mathbf{x}_i + \lambda_0) \geq \gamma \, ||\boldsymbol{\lambda}||_2 \quad i = 1 \dots n.$$

For any $\boldsymbol{\lambda}$ and λ_0 that satisfy this, any positively scaled multiple satisfies them too, so we can arbitrarily set $||\boldsymbol{\lambda}||_2 = 1/\gamma$ so that the right side is 1.

Now when we maximize γ, we're maximizing $\gamma = 1/\left|\left|\boldsymbol{\lambda}\right|\right|_2$. So we have

$$\max_{\boldsymbol{\lambda},\lambda_0} \frac{1}{||\boldsymbol{\lambda}||_2} \quad \text{s.t.} \quad y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) \geq 1 \quad i = 1 \dots n.$$

Equivalently,

$$\min_{\boldsymbol{\lambda},\lambda_0} \frac{1}{2} ||\boldsymbol{\lambda}||_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) - 1 \geq 0 \quad i = 1 \dots n \tag{1}$$

(the 1/2 and square are just for convenience) which is the same as:

$$\min_{\boldsymbol{\lambda},\lambda_0} \frac{1}{2} ||\boldsymbol{\lambda}||_2^2 \quad \text{s.t.} \quad -y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) + 1 \leq 0 \quad i = 1 \dots n$$

leading to the Lagrangian

$$\mathcal{L}\left([\boldsymbol{\lambda}, \lambda_0], \boldsymbol{\alpha}\right) = \frac{1}{2}\sum_{j=1}^{p} \lambda_j^2 + \sum_{i=1}^{n} \alpha_i \left[-y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) + 1\right]$$

Writing the KKT conditions, starting with Lagrangian stationarity, where we need to find the gradient wrt $\boldsymbol{\lambda}$ and the derivative wrt λ_0:

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}\left([\boldsymbol{\lambda}, \lambda_0], \boldsymbol{\alpha}\right) = \boldsymbol{\lambda} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \mathbf{0} \Longrightarrow \boldsymbol{\lambda} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

$$\frac{\partial}{\partial \lambda_0}\mathcal{L}\left([\boldsymbol{\lambda}, \lambda_0], \boldsymbol{\alpha}\right) = -\sum_{i=1}^{n} \alpha_i y_i = 0 \Longrightarrow \sum_{i=1}^{n} \alpha_i y_i = 0.$$

$$\begin{aligned}
\alpha_i \geq 0 \quad \forall i & \qquad \text{(dual feasibility)} \\
\alpha_i \left[-y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) + 1\right] = 0 \quad \forall i & \qquad \text{(complementary slackness)} \\
-y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) + 1 \leq 0. & \qquad \text{(primal feasibility)}
\end{aligned}$$

Using the KKT conditions, we can simplify the Lagrangian in order to get a nice

expression for the dual objective.

$$\mathcal{L}\left([\boldsymbol{\lambda}, \lambda_0], \boldsymbol{\alpha}\right) = \frac{1}{2}\|\boldsymbol{\lambda}\|_2^2 + \boldsymbol{\lambda}^T \sum_{i=1}^{n}(-\alpha_i y_i \mathbf{x}_i) + \sum_{i=1}^{n}(-\alpha_i y_i \lambda_0) + \sum_{i=1}^{n}\alpha_i$$

(We just expanded terms. Now we'll plug in the first KKT condition.)

$$= \frac{1}{2}\left|\left|\boldsymbol{\lambda}\right|\right|_2^2 - \left|\left|\boldsymbol{\lambda}\right|\right|_2^2 - \lambda_0 \sum_{i=1}^{n}(\alpha_i y_i) + \sum_{i=1}^{n}\alpha_i$$

(Plug in the second KKT condition.)

$$= -\frac{1}{2}\sum_{j=1}^{p}\lambda_j^2 \quad + \qquad 0 \qquad + \sum_{i=1}^{n}\alpha_i. \tag{2}$$

Again using the first KKT condition, we can rewrite the first term.

$$\begin{aligned} -\frac{1}{2}\sum_{j=1}^{p}\lambda_j^2 &= -\frac{1}{2}\sum_{j=1}^{p}\left(\sum_{i=1}^{n}\alpha_i y_i x_{ij}\right)^2 \\ &= -\frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i\alpha_k y_i y_k x_{ij} x_{kj} \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i\alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k. \end{aligned}$$

Plugging back into the Lagrangian (2), which now only depends on $\boldsymbol{\alpha}$, and putting in the second and third KKT conditions gives us the dual problem;

$$\max_{\boldsymbol{\alpha}} \mathcal{L}\left(\boldsymbol{\alpha}\right)$$

where

$$\mathcal{L}\left(\boldsymbol{\alpha}\right) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,k}\alpha_i\alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k \quad \text{s.t.} \quad \begin{cases} \alpha_i \geq 0 \quad i = 1 \ldots n \\ \sum_{i=1}^{n}\alpha_i y_i = 0 \end{cases} \tag{3}$$

We'll use the last two KKT conditions in what follows, for instance to get conditions on λ_0, but what we've already done is enough to define the dual problem for $\boldsymbol{\alpha}$.

We can solve this dual problem. Either (i) we'd use a generic quadratic programming solver, or (ii) use another algorithm, like SMO, which I will discuss

later. For now, assume we solved it. So we have $\alpha_1^*, \ldots, \alpha_n^*$. We can use the solution of the dual problem to get the solution of the primal problem. We can plug $\boldsymbol{\alpha}^*$ into the first KKT condition to get

$$\boldsymbol{\lambda}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i. \tag{4}$$

We still need to get λ_0^*, but we can see something cool in the process.

Support Vectors

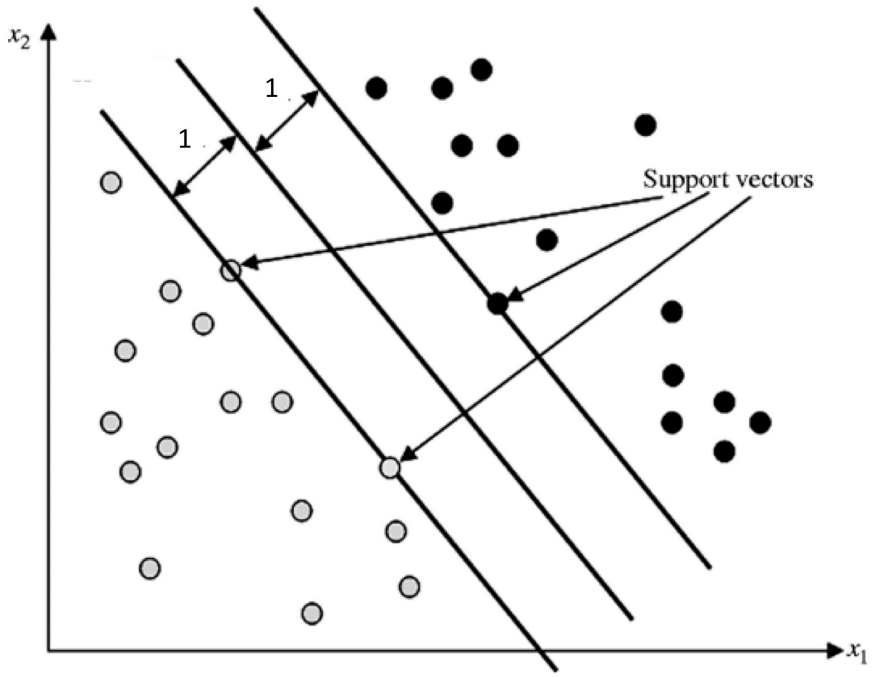
Look at the complementary slackness KKT condition and the primal and dual feasibility conditions:

$$\alpha_i^* \left[-y_i(\boldsymbol{\lambda}^{*T}\mathbf{x}_i + \lambda_0^*) + 1\right] = 0 \Rightarrow \begin{cases} \alpha_i^* > 0 \Rightarrow y_i(\boldsymbol{\lambda}^{*T}\mathbf{x}_i + \lambda_0^*) = 1 \\ \alpha_i^* < 0 \text{ (Can't happen)} \\ -y_i(\boldsymbol{\lambda}^{*T}\mathbf{x}_i + \lambda_0^*) + 1 < 0 \Rightarrow \alpha_i^* = 0 \\ -y_i(\boldsymbol{\lambda}^{*T}\mathbf{x}_i + \lambda_0^*) + 1 > 0 \text{ (Can't happen)} \end{cases}$$

Define the optimal (scaled) scoring function: $f^*(\mathbf{x}_i) = \boldsymbol{\lambda}^{*T}\mathbf{x}_i + \lambda_0^*$, then

$$\begin{cases} \alpha_i^* > 0 & \Rightarrow \quad y_i f^*(\mathbf{x}_i) = \text{scaled margin}_i = 1 \\ 1 < y_i f^*(\mathbf{x}_i) & \Rightarrow \quad \alpha_i^* = 0 \end{cases}$$

The observations in the first category, for which the scaled margin is 1 and the constraints are active are called *support vectors*. They are the closest to the decision boundary.



Finish What We Were Doing Earlier

To get λ_0^*, use the complementarity condition for any of the support vectors (in other words, use the fact that the unnormalized margin of the support vectors is one):

$$1 = y_i(\boldsymbol{\lambda}^{*T}\mathbf{x}_i + \lambda_0^*).$$

If you take a positive support vector, $y_i = 1$, then

$$\lambda_0^* = 1 - \boldsymbol{\lambda}^{*T}\mathbf{x}_i.$$

Written another way, since the support vectors have the smallest margins,

$$\lambda_0^* = 1 - \min_{i:y_i=1} \boldsymbol{\lambda}^{*T}\mathbf{x}_i.$$

So that's the solution! Just to recap, to get the scoring function f^* for SVM, you'd compute $\boldsymbol{\alpha}^*$ from the dual problem (3), plug it into (4) to get $\boldsymbol{\lambda}^*$, plug that into the equation above to get λ_0^*, and that's the solution to the primal problem, and the coefficients for f^*.

Because of the form of the solution:

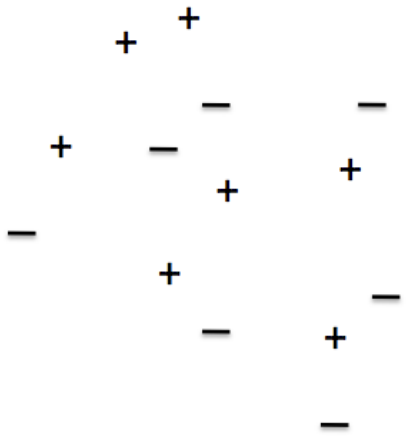
$$\boldsymbol{\lambda}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i.$$

it is possible that $\boldsymbol{\lambda}^*$ is very fast to calculate.

Why is that? Think support vectors.

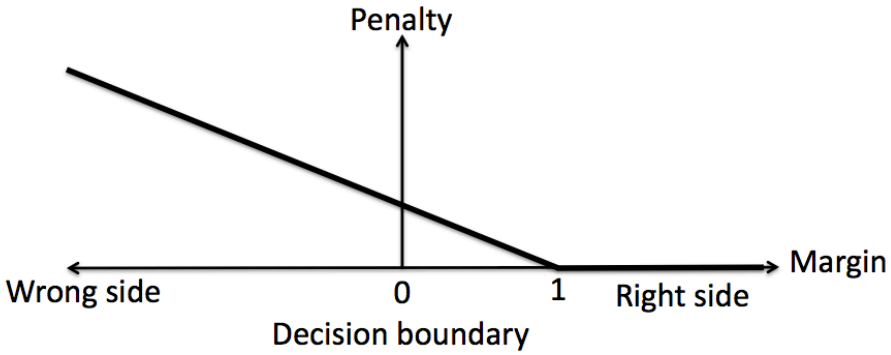
The Nonseparable Case

If there is no separating hyperplane,



there is no feasible solution to the problem we wrote above. Most real problems are nonseparable.

Let's fix our SVM so it can accommodate the nonseparable case. The new formulation will penalize mistakes the farther they are from the decision boundary. So we are allowed to make mistakes now, but we pay a price.

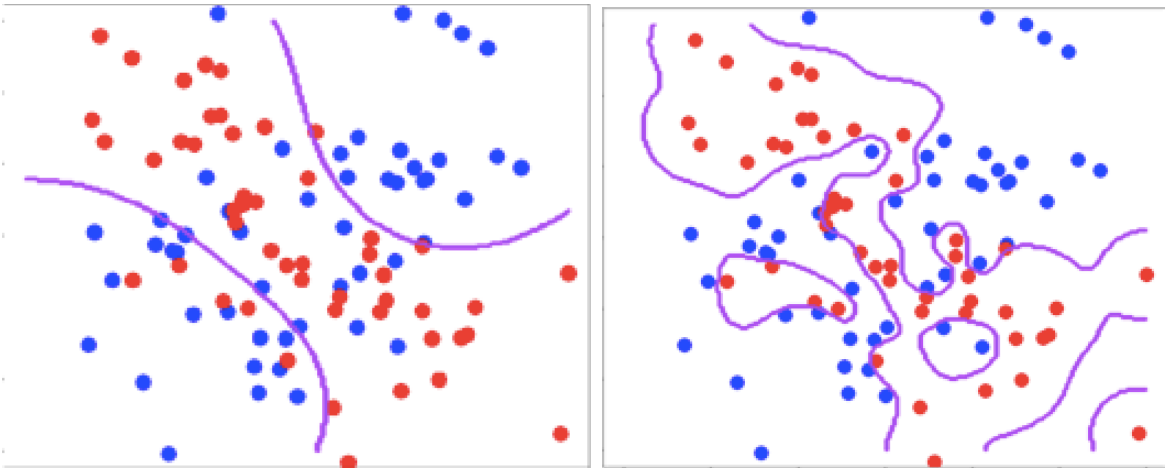


Let's change our primal problem (1) to this new primal problem:

$$\min_{\boldsymbol{\lambda},\lambda_0,\boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{\lambda}||_2^2 + C\sum_{i=1}^{n}\xi_i \quad \text{s.t.} \quad \begin{cases} y_i(\boldsymbol{\lambda}^T\mathbf{x}_i + \lambda_0) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \tag{5}$$

So the constraints allow some slack of size ξ_i, but we pay a price for it in the objective. That is, if $y_i f(\mathbf{x}_i) \geq 1$ then ξ_i gets set to 0, penalty is 0. Otherwise, if $y_i f(\mathbf{x}_i) = 1 - \xi_i$, we pay price ξ_i.

Parameter C trades off between the twin goals of making the $||\boldsymbol{\lambda}||_2^2$ small (making what-was-the-minimum-margin $1/||\boldsymbol{\lambda}||_2^2$ large) and ensuring that most observations have margin at least $1/||\boldsymbol{\lambda}||_2^2$. If you would choose to make the margins large at the expense of not all points being correctly classified, it tends to make the decision boundary smoother. These images below use kernels, which I haven't introduced yet, but hopefully you get the idea.



If you make C very large then it is the same as the separable case. Set C large. Then assuming there is a separable solution, we would have all the points

classified with a small but positive margin (which is the picture above on the right). If we decrease C, then we would tradeoff between the two (picture above on the left).

Going on a Little Tangent

Rewrite the penalty another way:
If $y_i f(\mathbf{x}_i) \geq 1$, zero penalty. Else, pay price $\xi_i = 1 - y_i f(\mathbf{x}_i)$

Third time's the charm:
Pay price $\xi_i = \lfloor 1 - y_i f(\mathbf{x}_i) \rfloor_+$
where this notation $\lfloor z \rfloor_+$ means take the maximum of z and 0.

Equation (5) becomes:

$$\min_{\boldsymbol{\lambda}, \lambda_0} \frac{1}{2} ||\boldsymbol{\lambda}||_2^2 + C \sum_{i=1}^{n} \lfloor 1 - y_i f(\mathbf{x}_i) \rfloor_+$$

Does that look familiar? It should!

The Dual for the Nonseparable Case

Form the Lagrangian of (5):

$$\mathcal{L}(\boldsymbol{\lambda}, \lambda_0, \xi, \alpha, r) = \frac{1}{2} ||\boldsymbol{\lambda}||_2^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left[y_i(\boldsymbol{\lambda}^T \mathbf{x}_i + \lambda_0) - 1 + \xi_i \right] - \sum_{i=1}^{n} r_i \xi_i$$

where α_i's and r_i's are Lagrange multipliers (constrained to be ≥ 0). The dual turns out to be (after some work)

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k \quad \text{s.t.} \quad \begin{cases} 0 \leq \alpha_i \leq C \quad i = 1 \ldots n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{6}$$

So the only difference from the original problem's Lagrangian (3) is that $0 \leq \alpha_i$ was changed to $0 \leq \alpha_i \leq C$. Neat!

(Note that the r_i's went away because one of the KKT conditions says they are $C - \alpha_i$.)

Solving the dual problem with SMO

SMO (Sequential Minimal Optimization) is a type of coordinate ascent algorithm, but adapted to SVM so that the solution always stays within the feasible region.

Start with (6). Let's say you want to hold $\alpha_2, \ldots, \alpha_n$ fixed and take a coordinate step in the first direction. That is, change α_1 to maximize the objective in (6). Can we make any progress? Can we get a better feasible solution by doing this?

Turns out, no. Look at the constraint in (6), $\sum_{i=1}^n \alpha_i y_i = 0$. This means:

$$\alpha_1 y_1 = -\sum_{i=2}^n \alpha_i y_i, \text{ or multiplying by } y_1,$$
$$\alpha_1 = -y_1 \sum_{i=2}^n \alpha_i y_i.$$

So, since $\alpha_2, \ldots, \alpha_n$ are fixed, α_1 is also fixed.

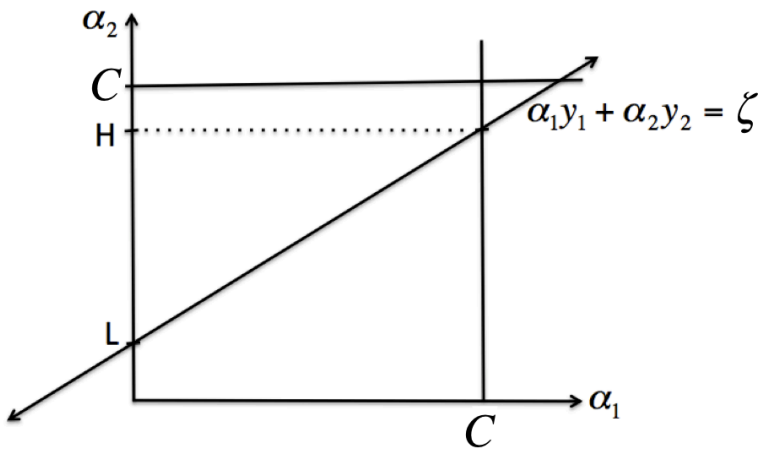
So, if we want to update any of the α_i's, we need to update at least 2 of them simultaneously to keep the solution feasible (i.e., to keep the constraints satisfied).

Start with a feasible vector $\boldsymbol{\alpha}$. Let's update α_1 and α_2, holding $\alpha_3, \ldots, \alpha_n$ fixed. What values of α_1 and α_2 are we allowed to choose?

Again, the constraint is: $\alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^n \alpha_i y_i =: \zeta$ (fixed constant).
We are only allowed to choose α_1, α_2 on the line, so when we pick α_2, we get α_1 automatically, from this:

$$\begin{aligned} \alpha_1 &= \frac{1}{y_1}(\zeta - \alpha_2 y_2) \\ &= y_1(\zeta - \alpha_2 y_2) \quad (y_1 = 1/y_1 \text{ since } y_1 \in \{+1, -1\}). \end{aligned}$$

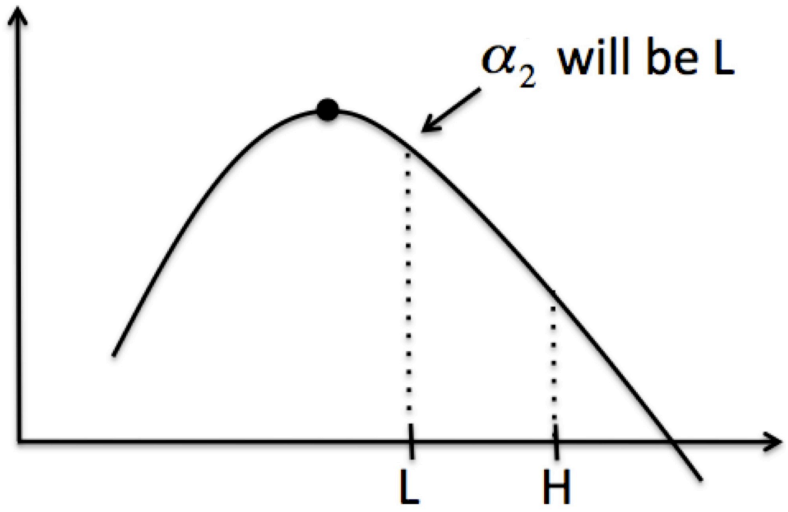
Also, the other constraints in (6) say $0 \leq \alpha_1, \alpha_2 \leq C$. So, α_2 needs to be within [L,H] on the figure (in order for α_1 to stay within $[0, C]$), where we will always have $0 \leq$ L,H $\leq C$. To do the coordinate ascent step, we will optimize the objective over α_2, keeping it within [L,H]. Intuitively, (6) becomes:



$$\max_{\alpha_2 \in [\mathrm{L},\mathrm{H}]} \left[\alpha_1 + \alpha_2 + \text{ constants } - \frac{1}{2} \sum_{i,k} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k \right] \text{ where } \alpha_1 = y_1(\zeta - \alpha_2 y_2). \tag{7}$$

The objective is quadratic in α_2. This means we can just set its derivative to 0 to optimize it and get α_2 for the next iteration of SMO. If the optimal value is outside of [L,H], just choose α_2 to be either L or H for the next iteration.

For instance, if this is a plot of (7)'s objective (sometimes it doesn't look like this, sometimes it's upside-down), then we'll choose :



Note: there are heuristics to choose the order of α_i's chosen to update.