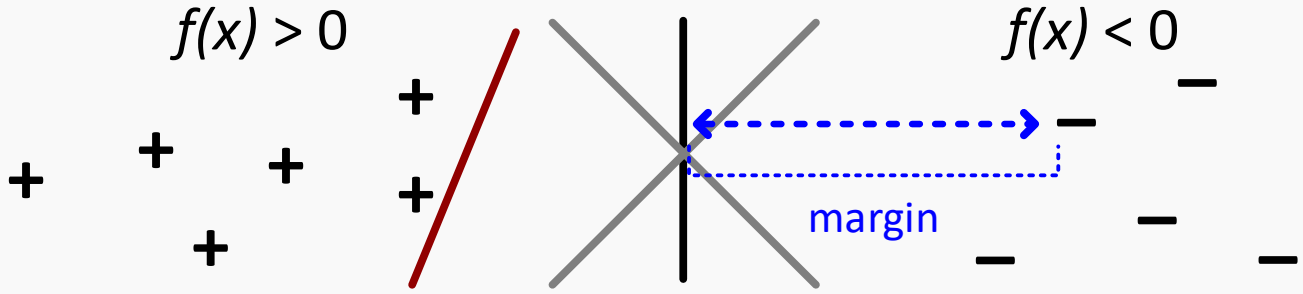
support vector machines ⇄ (SVM)

Support Vector Machines are very principles algorithms that can be applied in an elegant manner.

- They arose directly out of Statistical Learning Theory
- The utilize optimization techniques for efficient and structured application
- They leverage kernels to allow for powerful nonlinear classification problem application

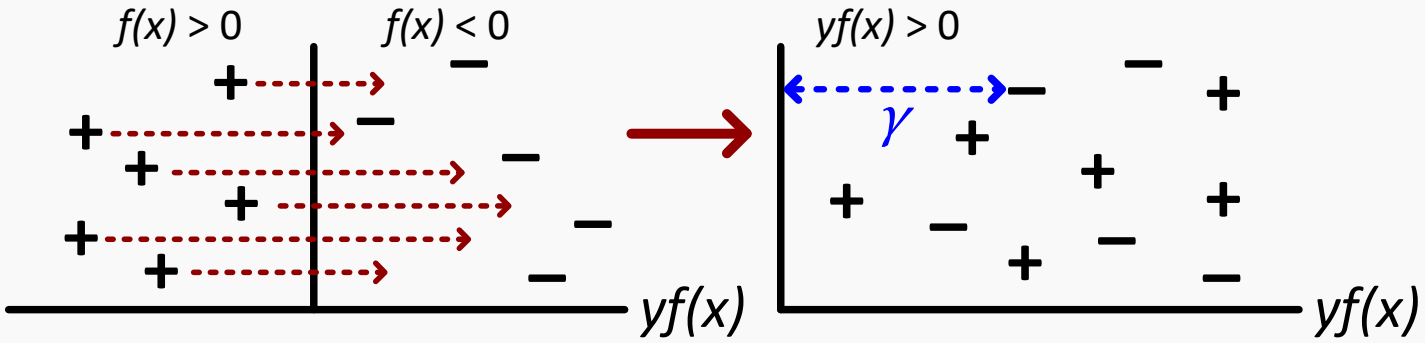
Support Vector Machines by addressing the questions of which **decision boundary** to select when all points in a dataset are classified correctly:



All of the above decision boundary assignments are equally plausible and equally 100% accurate. However, the **grey** boundaries and the **red** boundary are poor choices. They each fail at achieving the motivation behind **Support Vector Machines**.

The motivation behind **SVM** is to have all points as far away from the decision boundary as possible. As illustrated above, a decision boundary can be 100% accurate in classifying the points, but a poorly plotted decision boundary could cause a model to fail ay **generalization** to new data introduced.

The motivation above is achieved through examine the **margin** of the decision boundary; the distance between each data point and the decision boundary itself. In other words, the **minimum margin** in the model should ideally be **large**:



The **SVM optimization objective** is that all **points** $\boldsymbol{i}$ are farther than **gamma** $\boldsymbol{\gamma}$ from the decision **boundary**:

$y_i f(x_i) \geq \gamma, i = 1, \ldots, n$ → with the objective to **maxmimize** $\boldsymbol{\gamma}$

In formal terms, the objective is to **maximize** $\boldsymbol{\gamma}$, subject to the margins being **at least** $\boldsymbol{\gamma}$.

The first step to applying an **SVM**:

$\text{maximize } \boldsymbol{\gamma} \text{ such that } y_i f(x_i) \geq \boldsymbol{\gamma}, \quad i = 1, \ldots, n$

The issue existing with the first step is the ability to extract value from $\boldsymbol{\gamma}$ and $\boldsymbol{f(x_i)}$. The above step suffers from a scaling issue illustrated in example below:

$$y_i f(x_i) \times 100{,}000 \geq \boldsymbol{\gamma} \times 100{,}000, \qquad i = 1, \dots, n$$

Multiplying either side by a large number results in both $\boldsymbol{\gamma}$ and $\boldsymbol{f}(\boldsymbol{x_i})$ being arbitrarily large. Consequentially, $\boldsymbol{\gamma}$ will be forced into maintaining a relative size of $\boldsymbol{f}$:

$$y_i f(x_i) \geq \boldsymbol{\gamma} \times \text{"size } f\text{"}, \qquad i = 1, \dots, n$$

Although gamma $\boldsymbol{\gamma}$ is now meaningful, it remains arbitrary to which $\boldsymbol{f}$ is selected to apply. Meaning the equation will hold regardless of which equation $\boldsymbol{f}$ is selected ($\boldsymbol{2f}$ and $\boldsymbol{f}$ are equally good).

To address the arbitrary nature of f illustrated above, the expression will be set to 0:

$$y_i f(x_i) \geq \boldsymbol{\gamma} \times \text{size f} = 1 \;\longrightarrow \text{therefore } \boldsymbol{\gamma} = \frac{1}{\text{"size } f\text{"}}$$

The optimization objective after addressing the above issues:

$$\text{maximize } \gamma \text{ such that } y_i f(x_i) \geq \boldsymbol{\gamma} \times \text{size f} = 1$$

$$\text{maximize } \frac{1}{\text{"size } f\text{"}} \text{ such that } y_i f(x_i) \geq 1$$

The optimization problem now states to maximize 1 over the value of the size of f with the constraint that the minimum margin value is at least 1.

In regard to the function $f(x_i)$, a linear model will be chosen:

$$f(x_i) = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

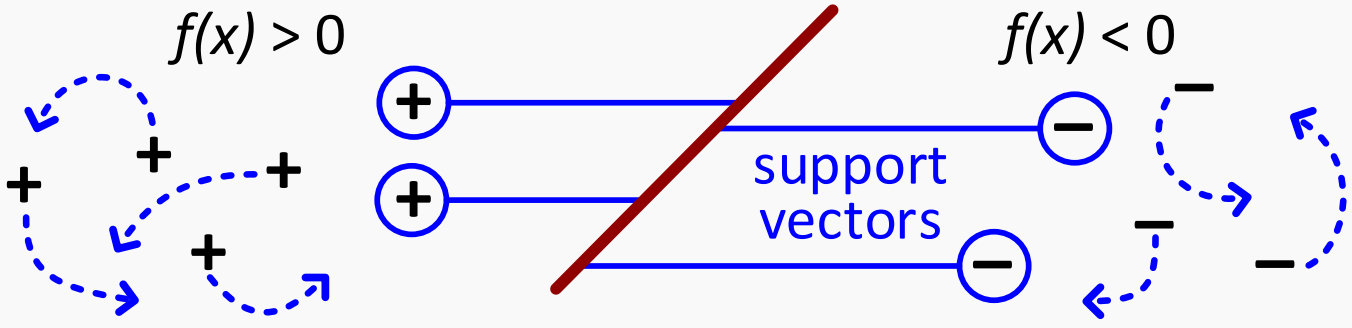
$$\text{size f} = \beta_1^2 + \beta_2^2 + \cdots + \beta_2^p = \|\beta\|_2^2 \longrightarrow \ell_2 \text{ normailization}$$

Applying the function f to the **SVM** objective:

$$\text{maximize } \frac{1}{\|\beta\|_2^2} \text{ such that } y_i f(x_i) \geq 1$$

$$\textbf{minimize } \|\beta\|_2^2 \text{ such that } y_i f(x_i) \geq 1$$

The **Support Vector Machine** optimization problem is solved through standard techniques in convex optimization (Lagrange multipliers, KKT conditions). Fortunately, the machinery designed to handle problems as such is fairly standard, with a wide array of toolboxes available specifically for optimization problems like **Support Vector Machines**.



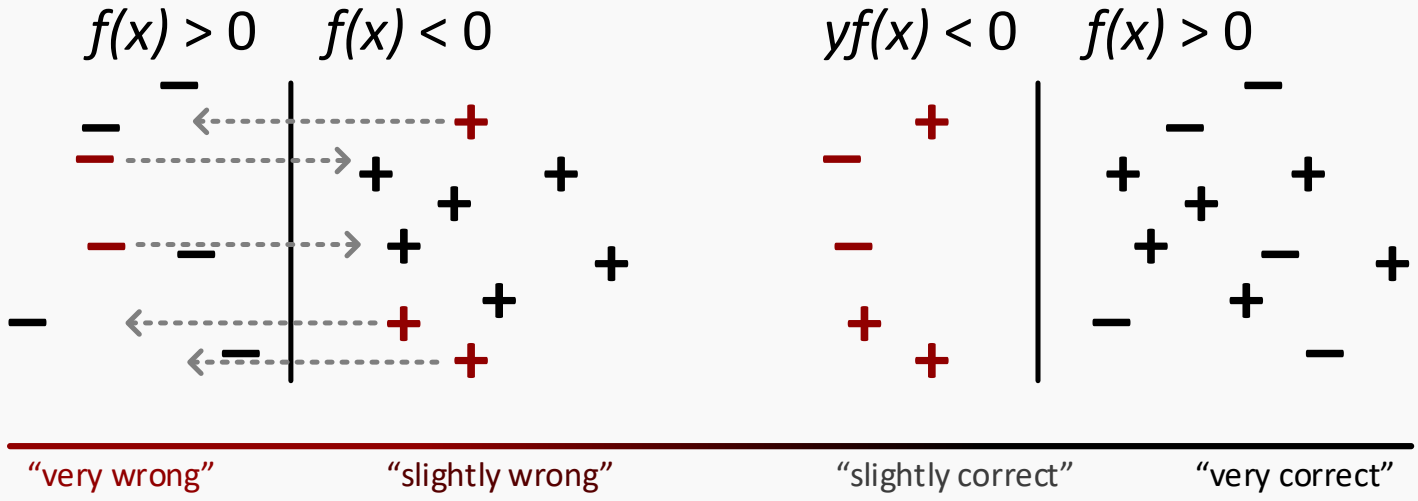
An important and interesting property regarding the solution to a Support Vector Optimization Problem is that only certain points will end up in the final solution for computing the β's. Meaning shifting the points on either side of the decision boundary around will have **no effect** on the final

solution. The special points that create the final solution are referred to as the **Support Vectors**. The **Support Vectors** are thus the points that are closest to the decision boundary on either side.

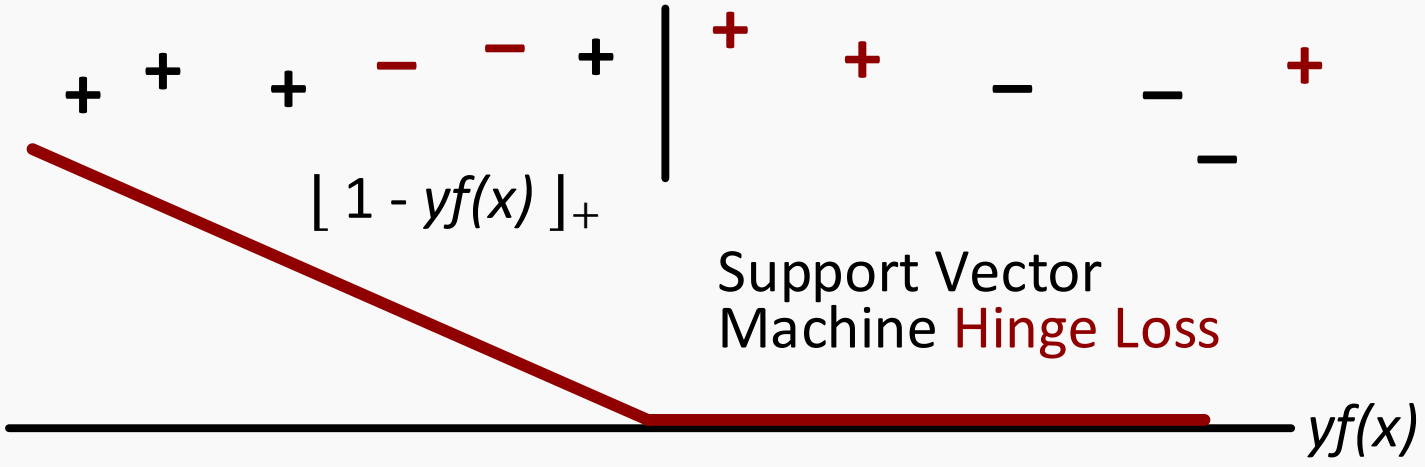
The Nonseparable Case

The problem illustrated above focused on all cases where the dataset can be separated perfectly by a given linear decision boundary. However, it is common for the dataset to contain much noise:

Returning to the geometric illustration familiar from before where the misclassified points are mapped to one side and the correctly classified points mapped to the other:



With the above understanding, it is necessary to implement that logic into the **SVM** function with the purpose of penalizing misclassified points more severely as they become farther away from the decision boundary they are misclassified with. The latter is achieved through the **SVM Hinge Loss**:



The intuition behind the above illustration of the **SVM Hinge Loss** is that when a point is correctly classified, there is no penalty; when a point is incorrectly classified, it is penalized in proportion to the distance away from the decision boundary. Therefore, the **more severe** the **misclassification** is, the heavier the respective penalty will be for the point. Note there is a slight overlap in the **Hinge Loss** function, indicating that a point can still suffer a slight amount of **loss** even it is correctly classified.

Applying the above **Loss Function** to the **Separable SVM** formulation:

$$\textbf{minimize } \|\beta\|_2^2 \text{ such that } y_i f(x_i) \geq 1$$

$$f(x_i) = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

Plugging in the **SVM Hinge Loss**:

$$\textbf{minimize } \|\beta\|_2^2 + C \sum_{i=1}^{n} [1 - y f(x)]_+$$

$$f(x_i) = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

The above formulation for **Nonseparable Cases of Support Vector Machines** is simply a special case of the standard **Machine Learning method minimizing a mixture of loss and regularization**:

$$f^* = \underset{\text{models } f}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i f(x_i)\big) + \boldsymbol{C} + \text{Regulatization}(f)$$

$$f(x_i) = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots + \beta_p x_{ip}$$

$$\text{Regularization}(f) = \beta_1^2 + \beta_2^2 + \beta_3^2 + \cdots + \beta_p^2 = \|\beta\|_2^2 \text{ (referred to as } \ell_2)$$

In practice, the optimization problem is solved using **Sequential Minimal Optimization** (**SMO**), where two variables are adjusted at a time. This is similar to **Coordinate Descent** utilized in **Boosting**. The difference is that a single variable cannot be adjusted at a time; this would violate constraints. Adjusting two variables at a time allows for flexibility to accommodate the latter constraints.