Support Vector Machines

Support Vector Machines

- Support Vector Machines are one of the more complex algorithms we will learn, but it all begins with a simple premise:
    - Does a hyperplane exist that can effectively separate classes?

Support Vector Machines

- Section Overview
    - Intuition and Theory for SVM
    - SVM Classification Example
    - SVM Regression Example
    - SVM Project Exercise and Solutions

Support Vector Machines

- Relevant Reading in ISLR
    - Chapter 9 covers Support Vector Machine Classification
    - Wikipedia has a section on Support Vector Machine Regression.

Support Vector Machines

Theory and Intuition - Hyperplanes and Margins

Support Vector Machines

- We will slowly build up to SVMs:
    - Maximum Margin Classifier
    - Support Vector Classifier
    - Support Vector Machines
- Let’s begin by understanding what a **hyperplane** is...

Support Vector Machines

- In an N-dimensional space, a hyperplane is a flat affine subspace of hyperplane dimension N − 1.
- For example:
    - 1-D Hyperplane is a single point
    - 2-D Hyperplane is a line
    - 3-D Hyperplane is flat plane

Support Vector Machines

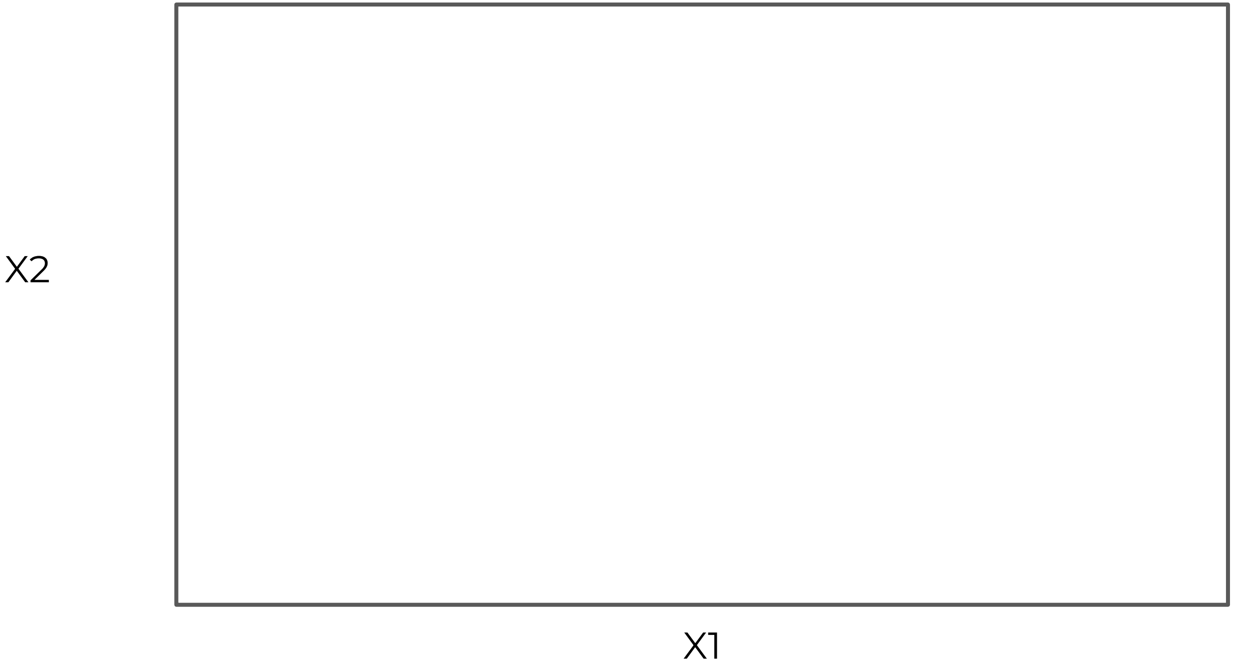
- 1-D Hyperplane is a single point

Support Vector Machines

- 1-D Hyperplane is a single point

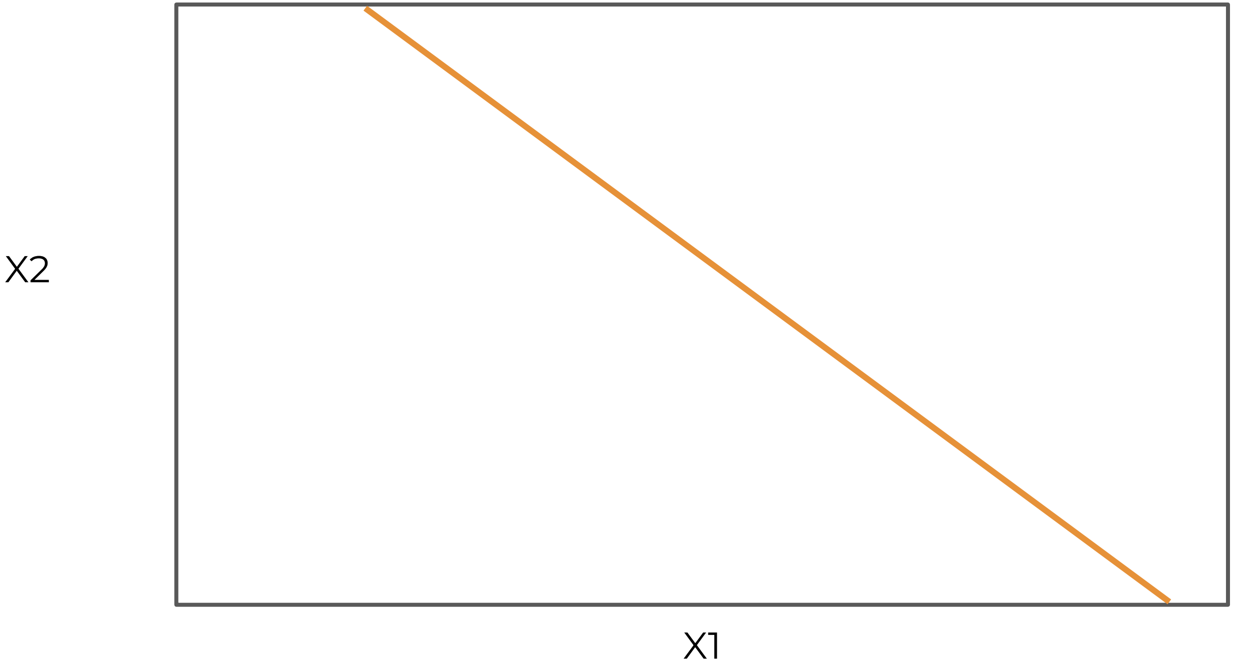
Support Vector Machines

- 2-D Hyperplane is a line



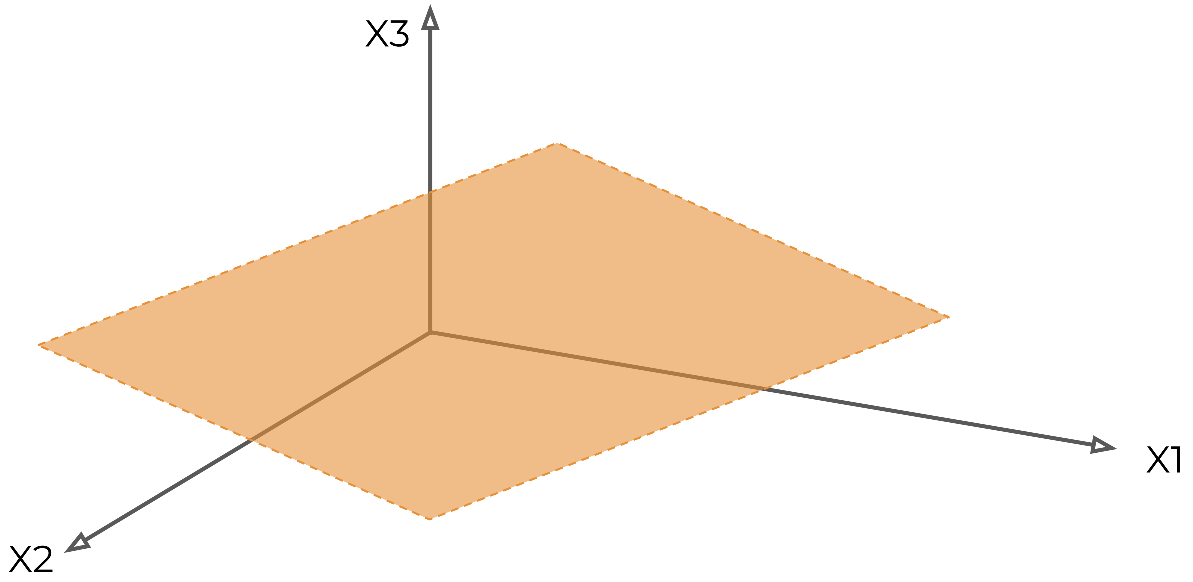
Support Vector Machines

- 2-D Hyperplane is a line



Support Vector Machines

- 3-D Hyperplane is a flat plane



Support Vector Machines

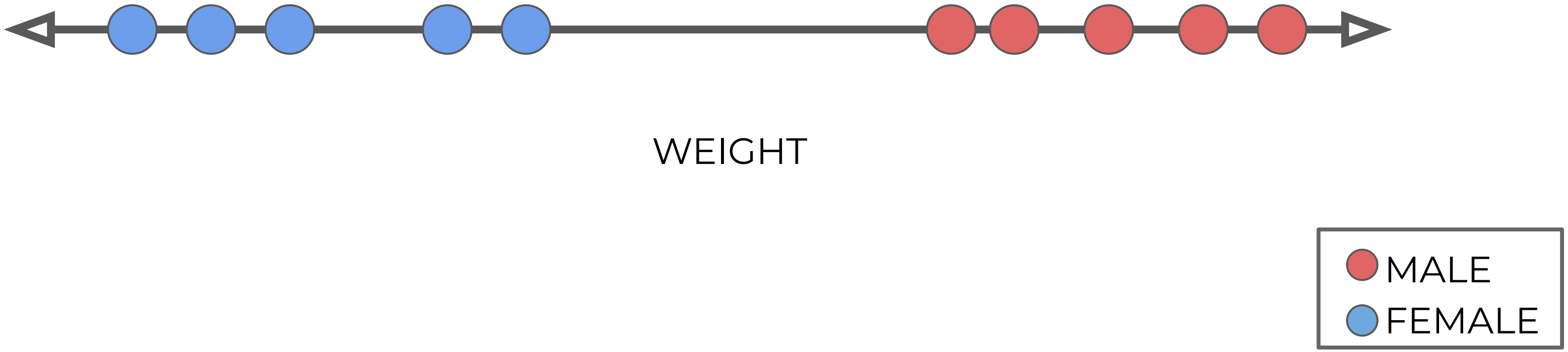
- The main idea behind SVM is that we can use Hyperplanes to create a separation between classes.
- Then new points will fall on one side of this separating hyperplane, which we can then use to assign a class.

Support Vector Machines

- Imagine a data set with one feature and one binary target label.
- For example:
    - A weight feature for baby chicks
    - Classified by Male or Female
- What would this look like visualized?

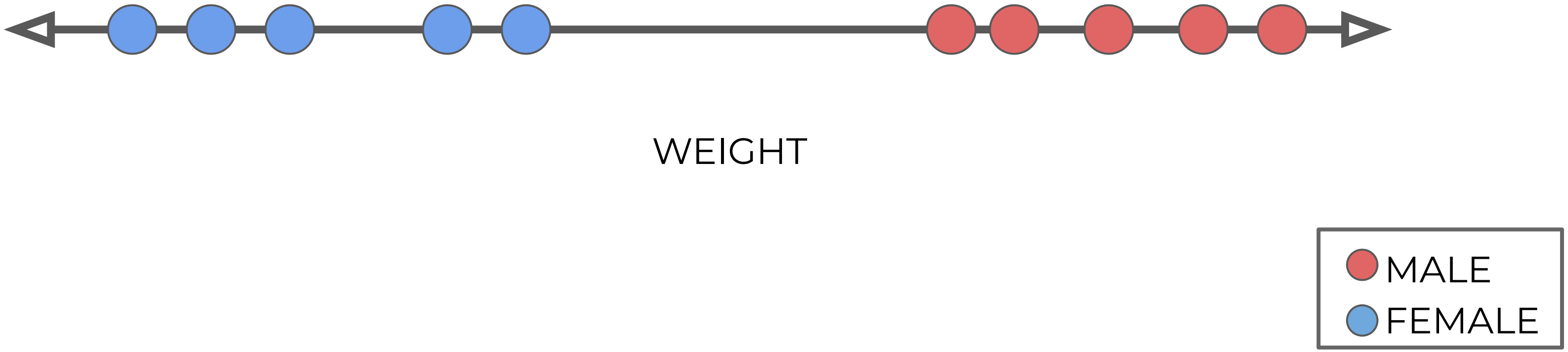
Support Vector Machines

- Place points along feature.



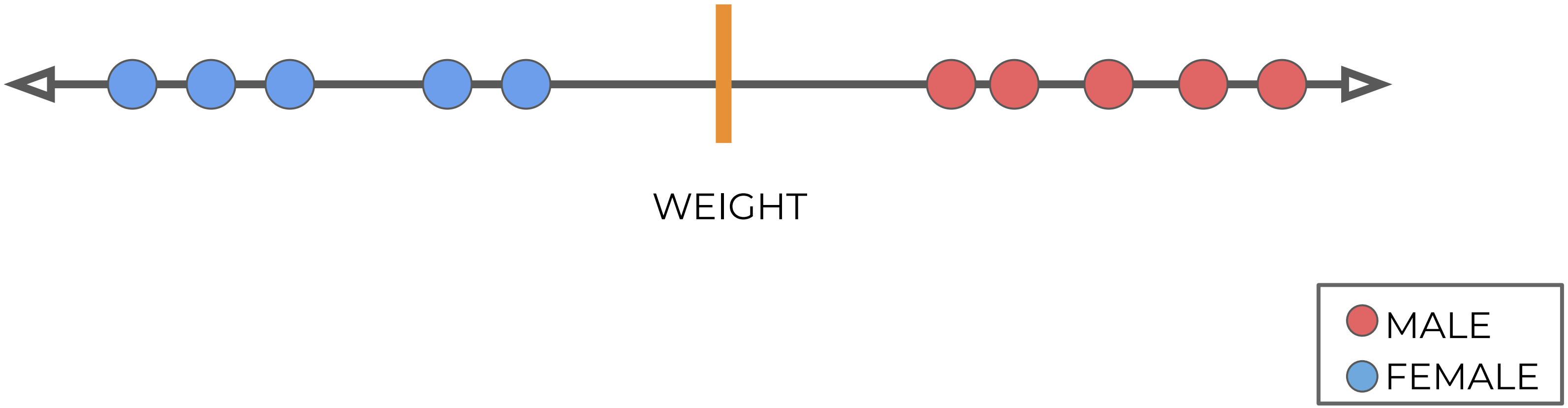
Support Vector Machines

- Notice in this case, classes are perfectly separable. This is unlikely in real world datasets.



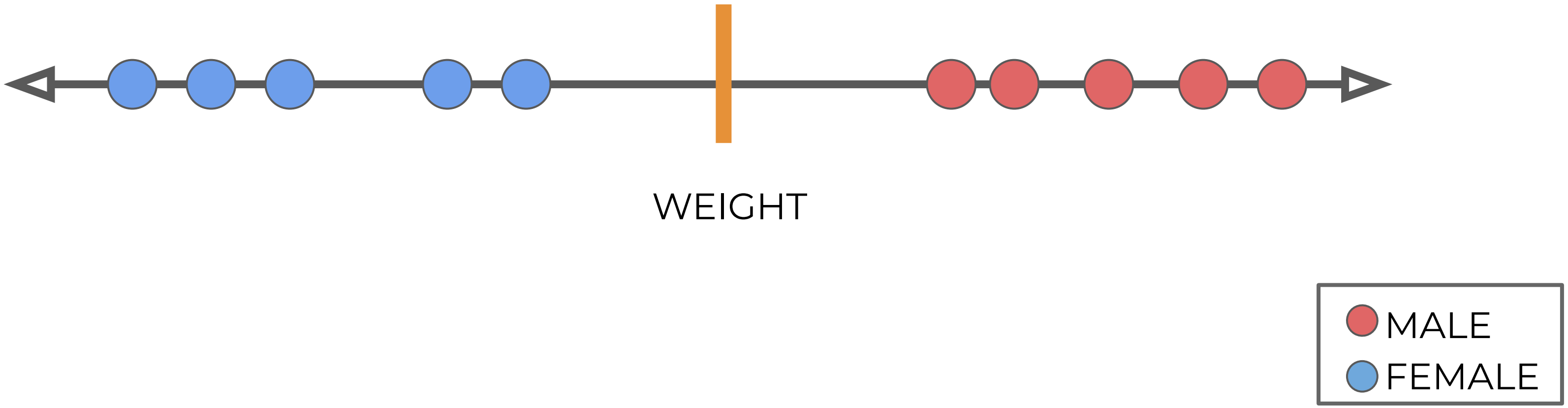
Support Vector Machines

- Idea behind SVM is to create a **separating hyperplane** between the classes.



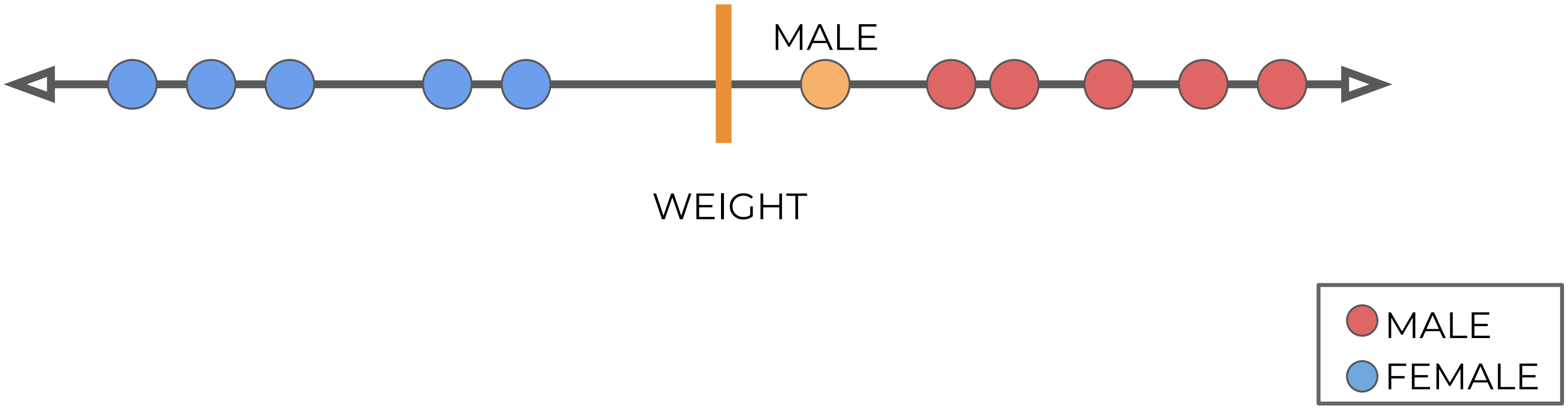
Support Vector Machines

- A new point would be classified based on what side of the point they land on.



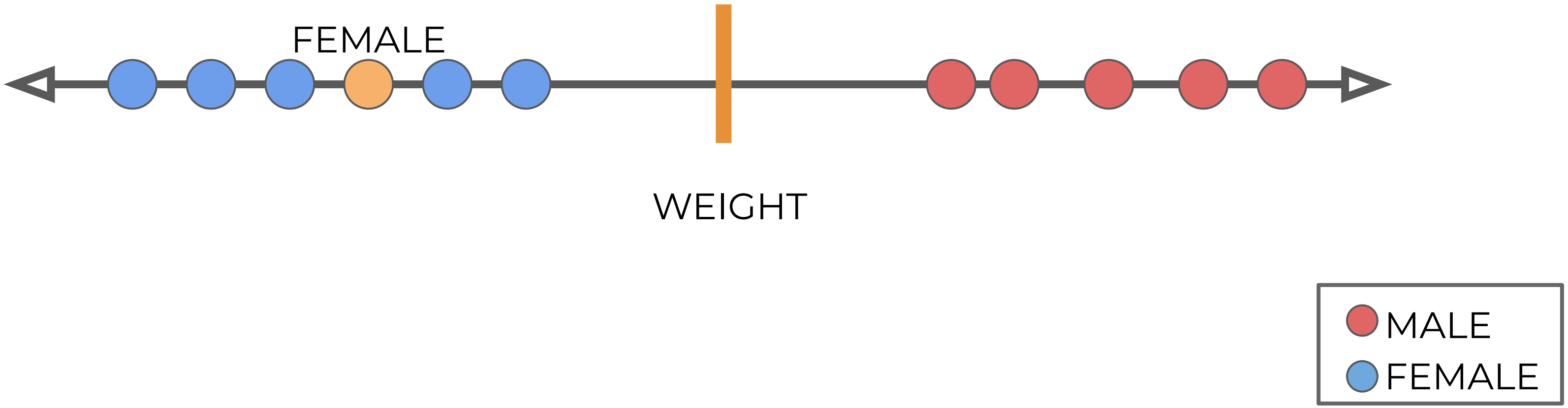
Support Vector Machines

- A new point would be classified based on what side of the point they land on.



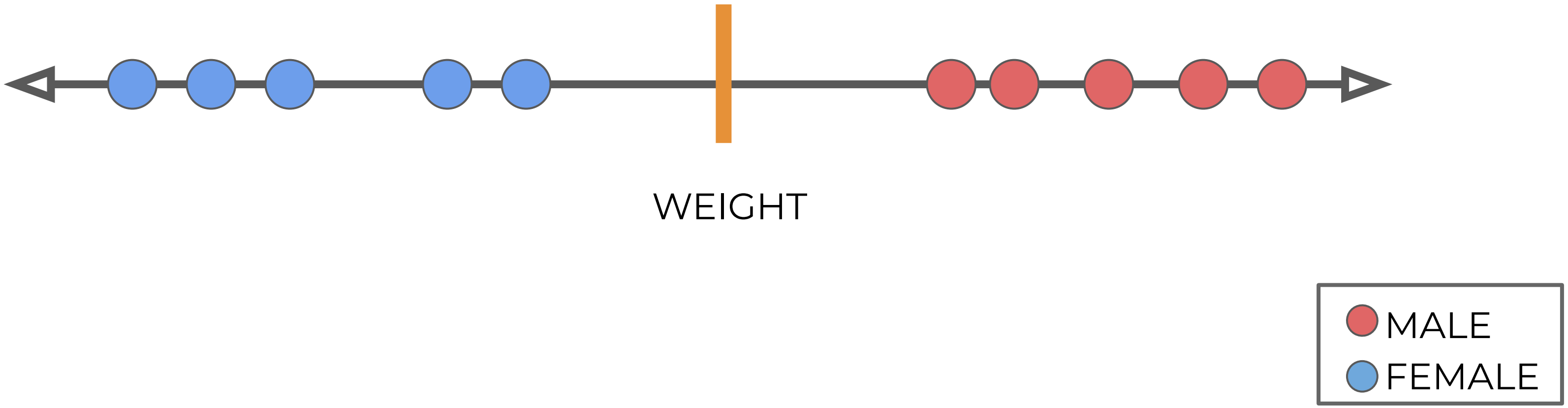
Support Vector Machines

- A new point would be classified based on what side of the point they land on.



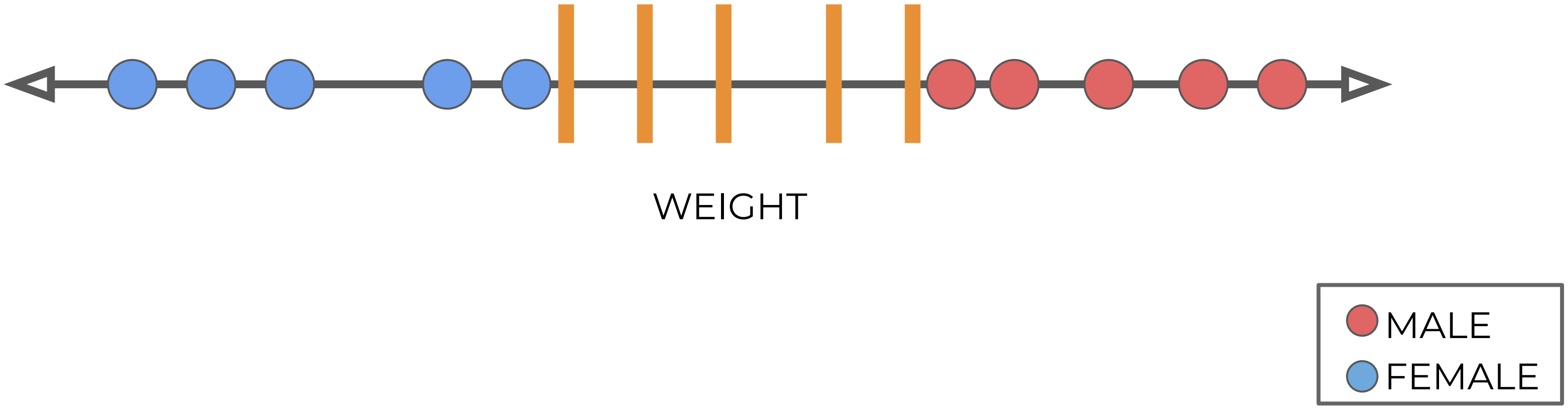
Support Vector Machines

- How do we choose where to put this separating hyperplane?



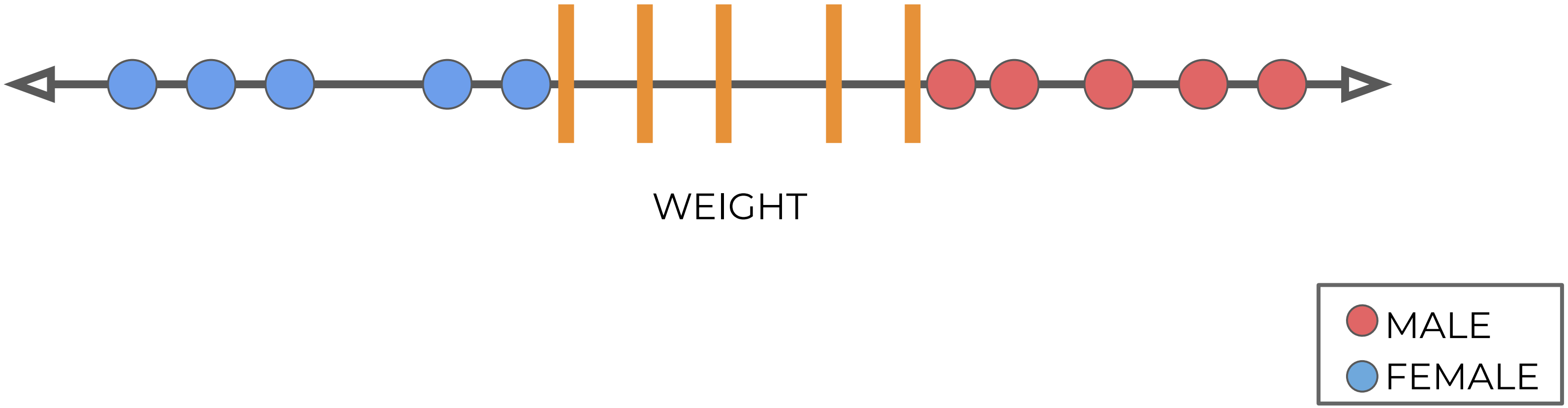
Support Vector Machines

- Note there are many options that perfectly separate out these classes:



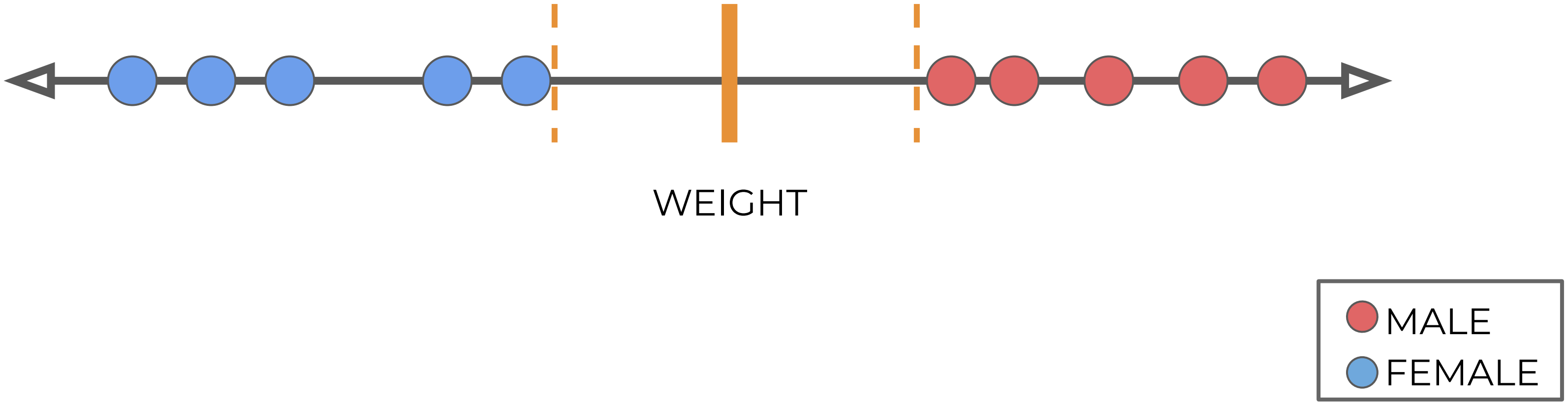
Support Vector Machines

- Which of these is the “best” separator between the classes?



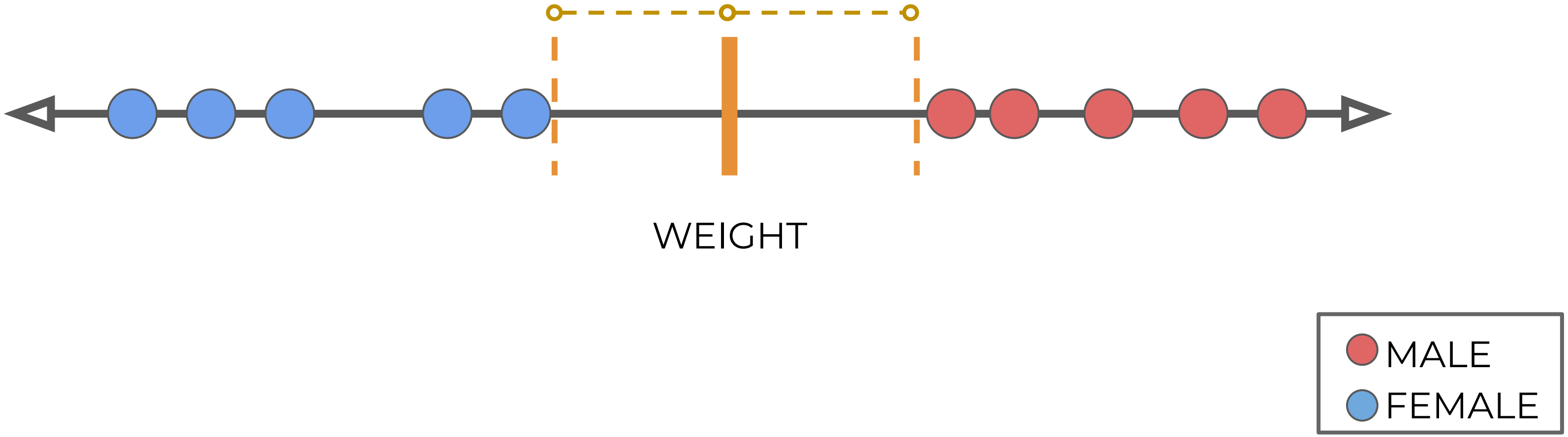
Support Vector Machines

- We could use the separator that **maximizes** the **margins** between the classes.



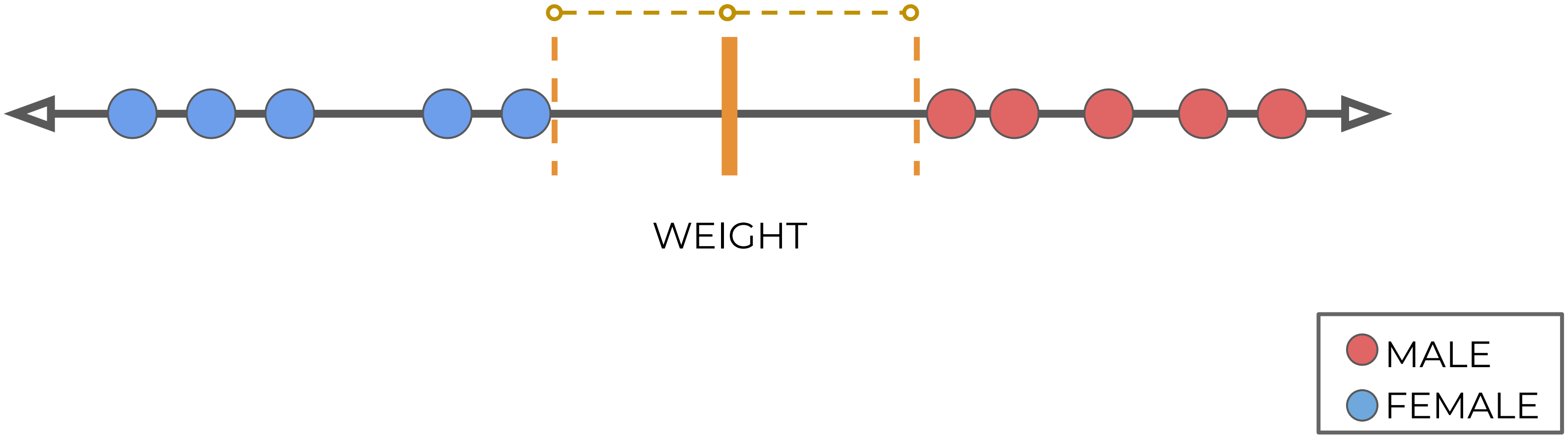
Support Vector Machines

- We could use the separator that **maximizes** the **margins** between the classes.



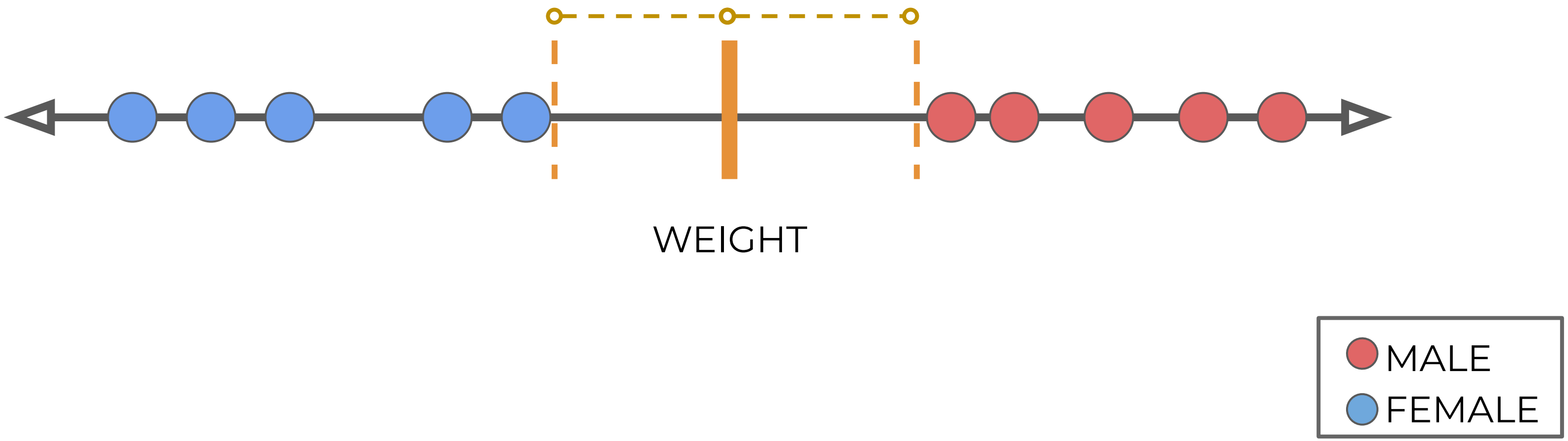
Support Vector Machines

- This is known as a **Maximal Margin Classifier.**



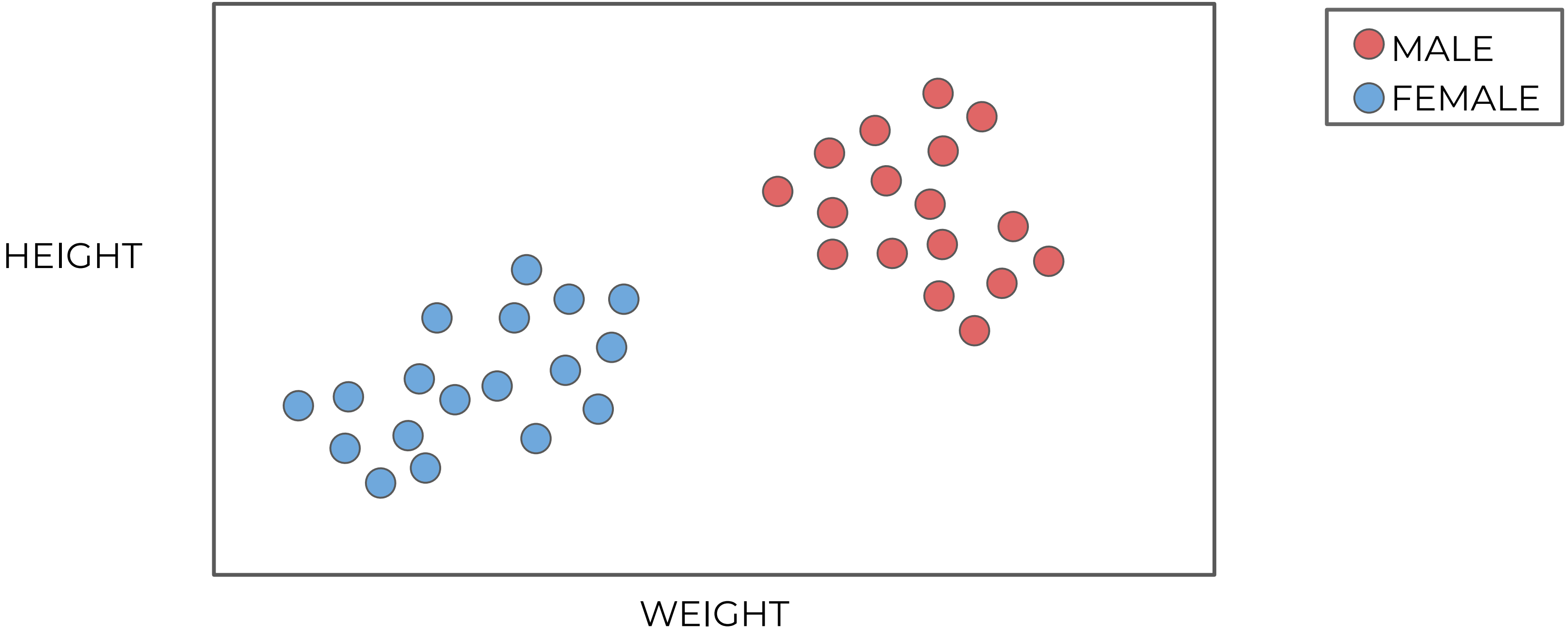
Support Vector Machines

- This same idea of maximum margins applies to N-dimensions.



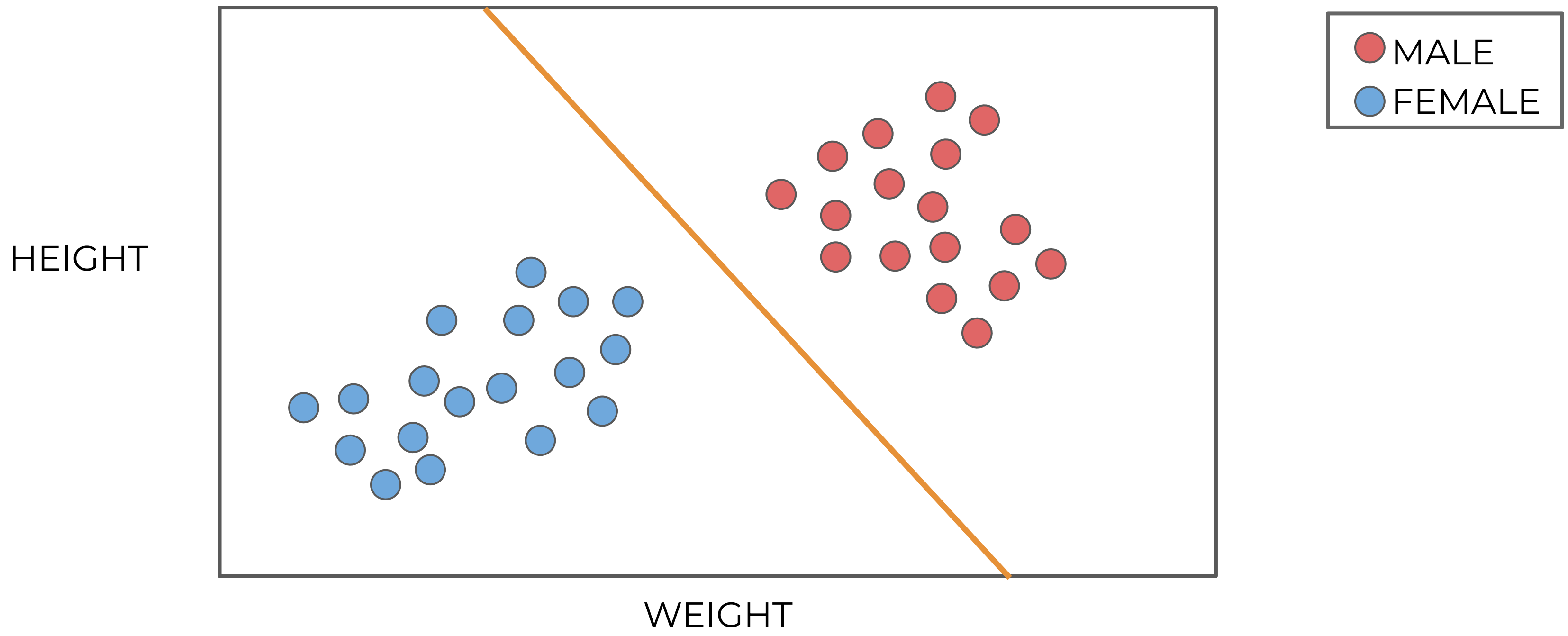
Support Vector Machines

- Imagine a 2 dimensional feature space:



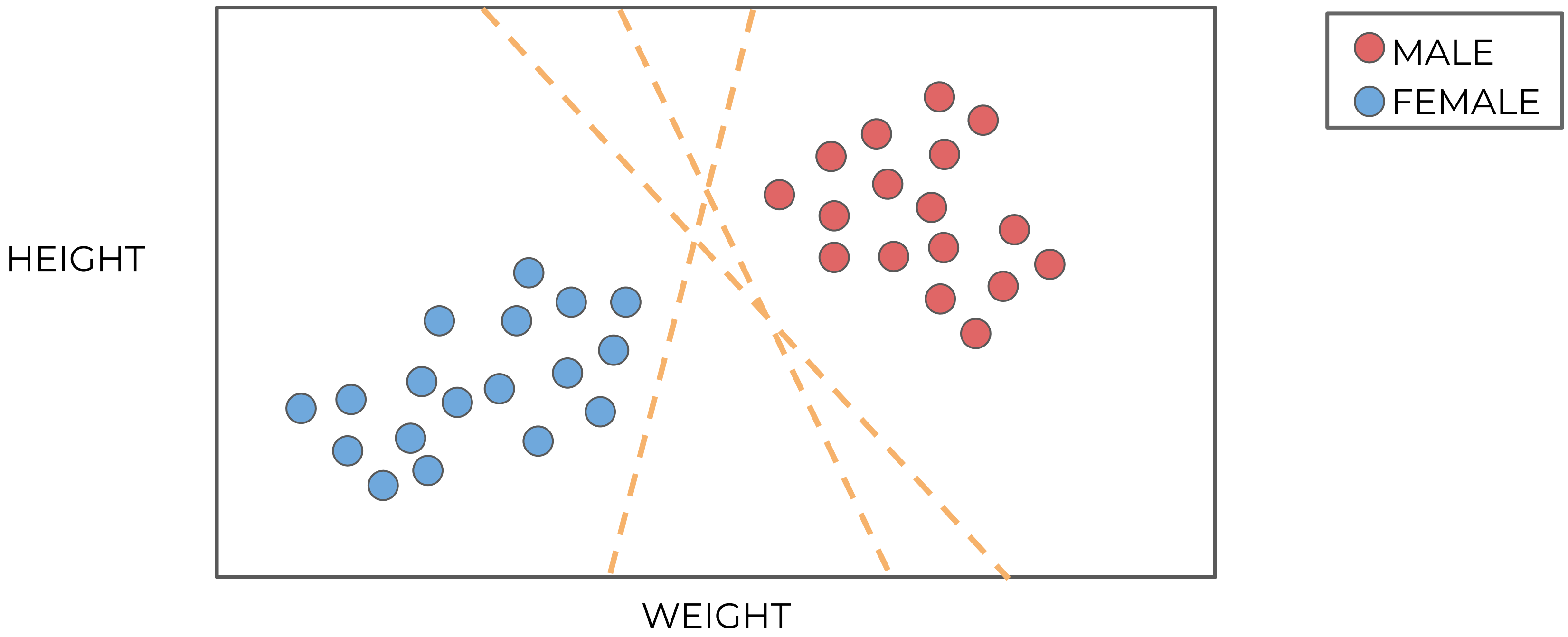
Support Vector Machines

- Separating hyperplane is a line.



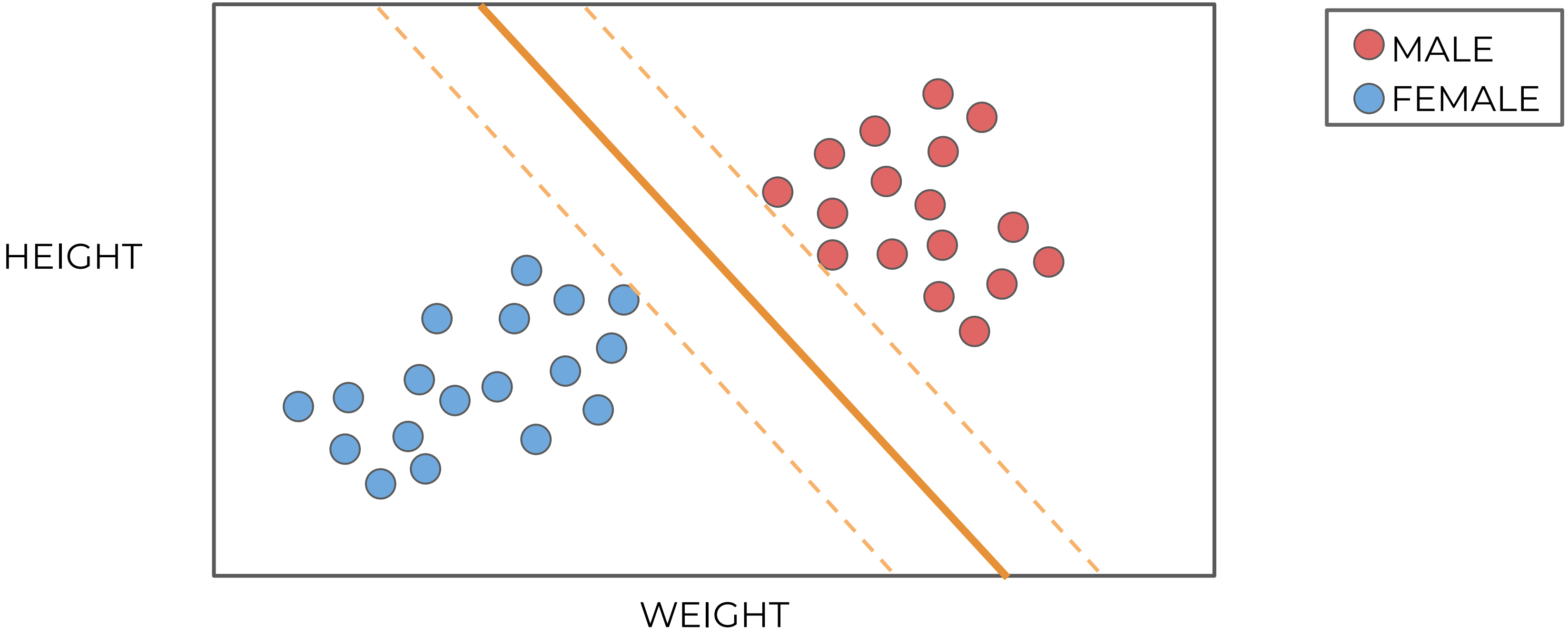
Support Vector Machines

- Multiple possible hyperplanes:



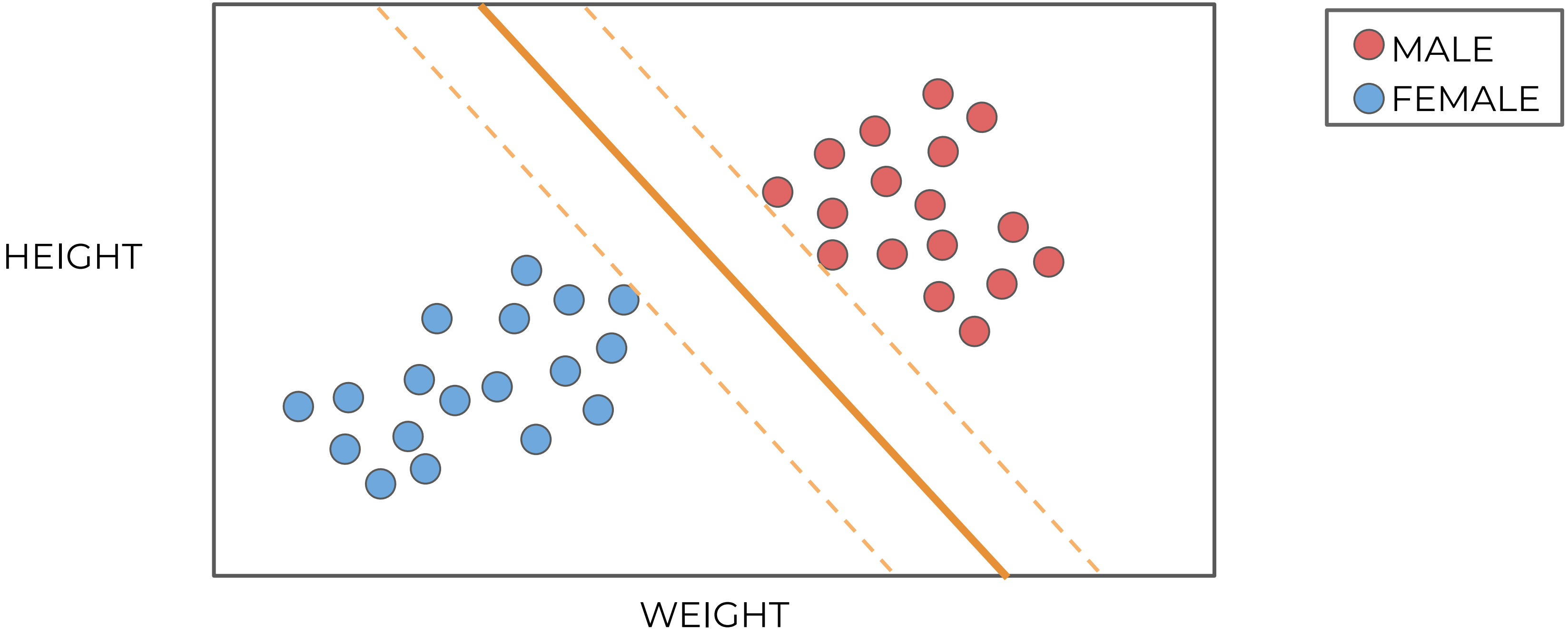
Support Vector Machines

- Choose to maximize margins:



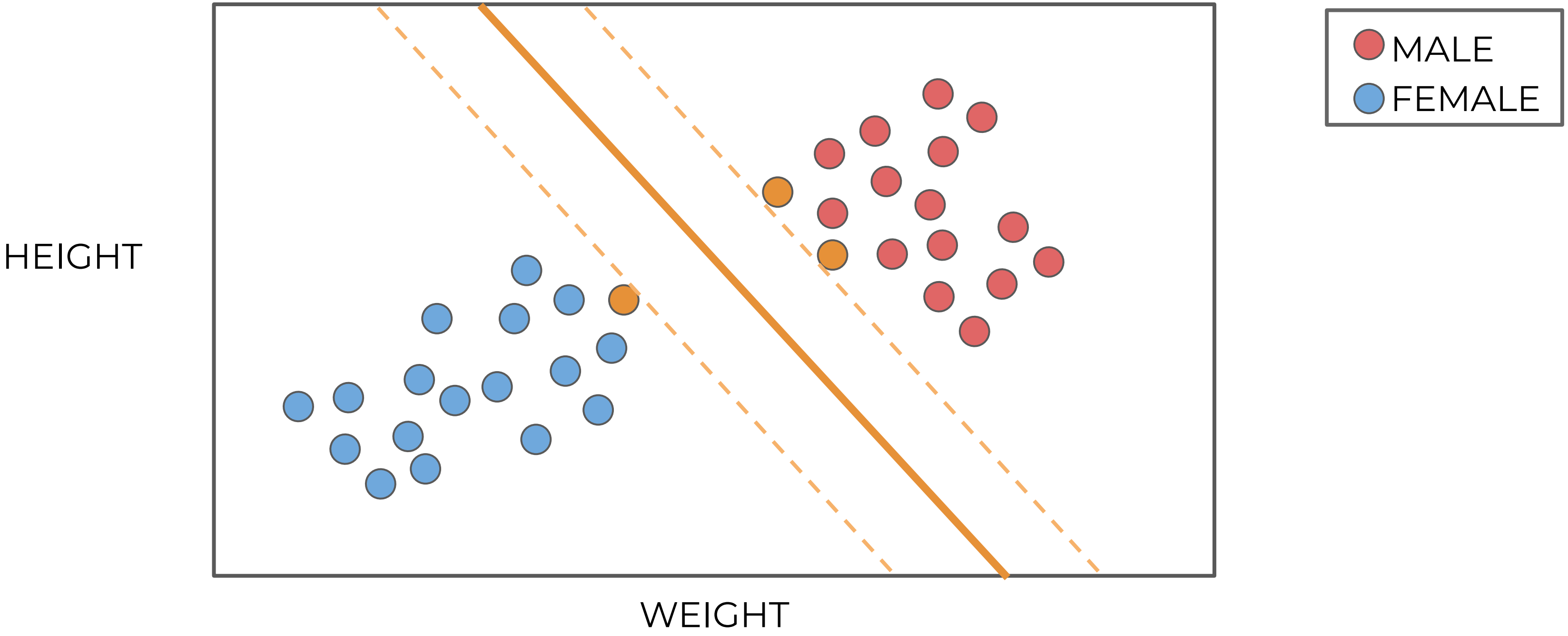
Support Vector Machines

- Note each data point is a 2D vector:



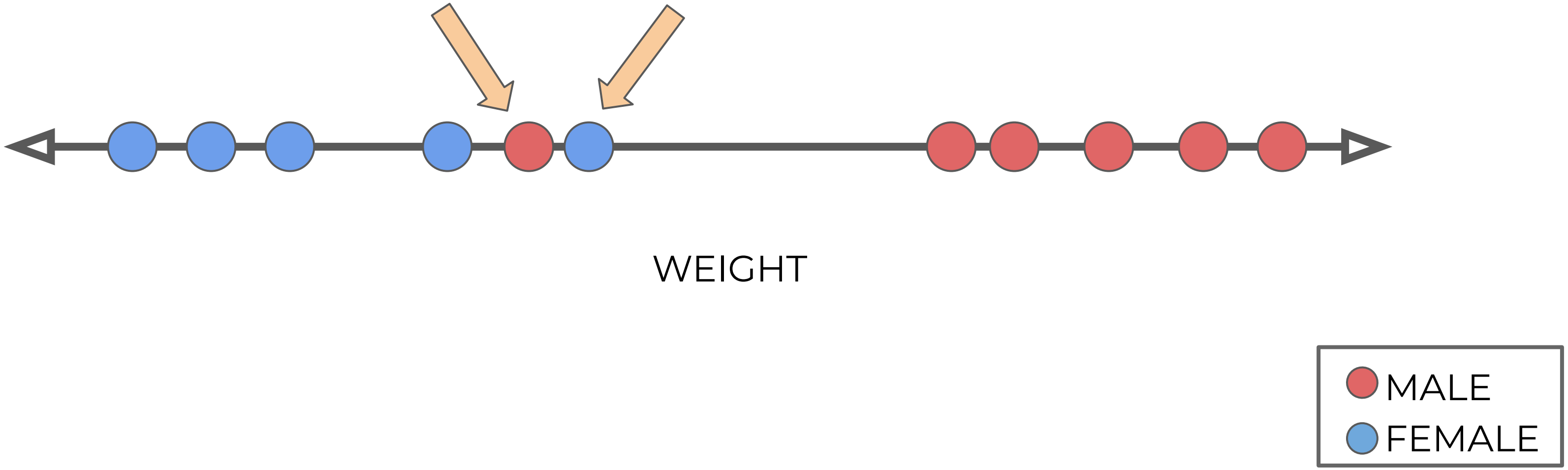
Support Vector Machines

- Data points at margin “support” separator:



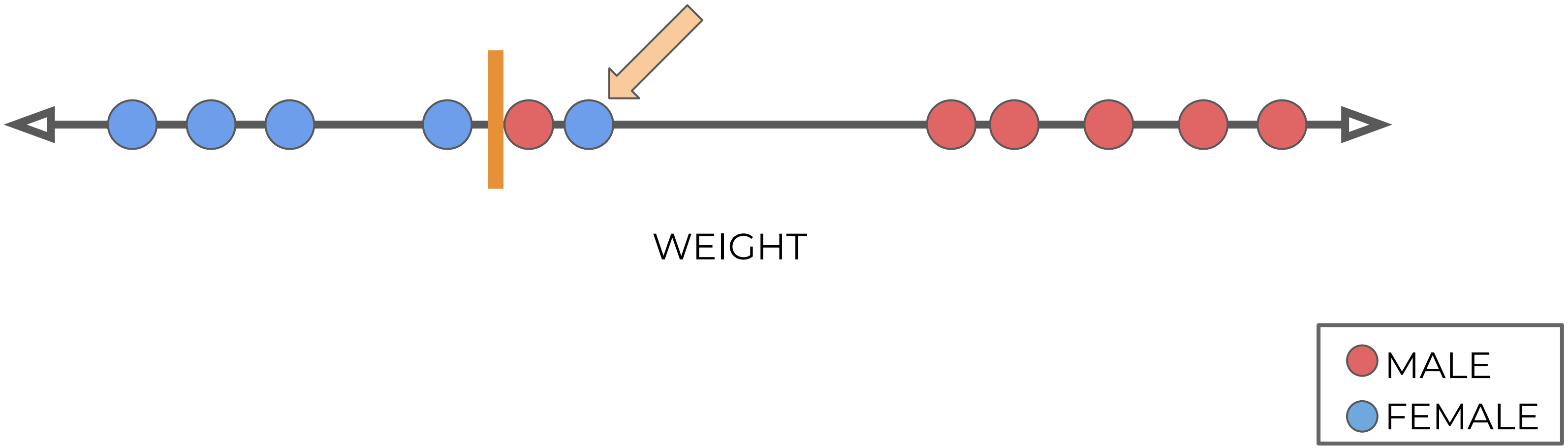
Support Vector Machines

- What happens if classes are not perfectly separable?



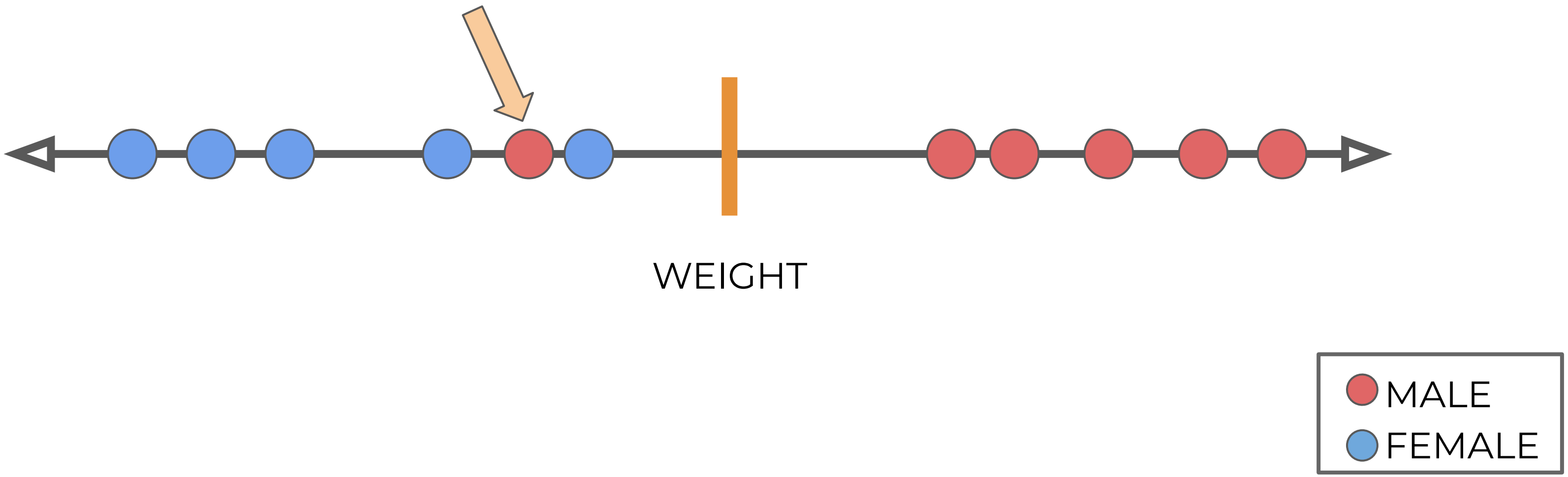
Support Vector Machines

- We are not be able to separate without allowing for misclassifications.



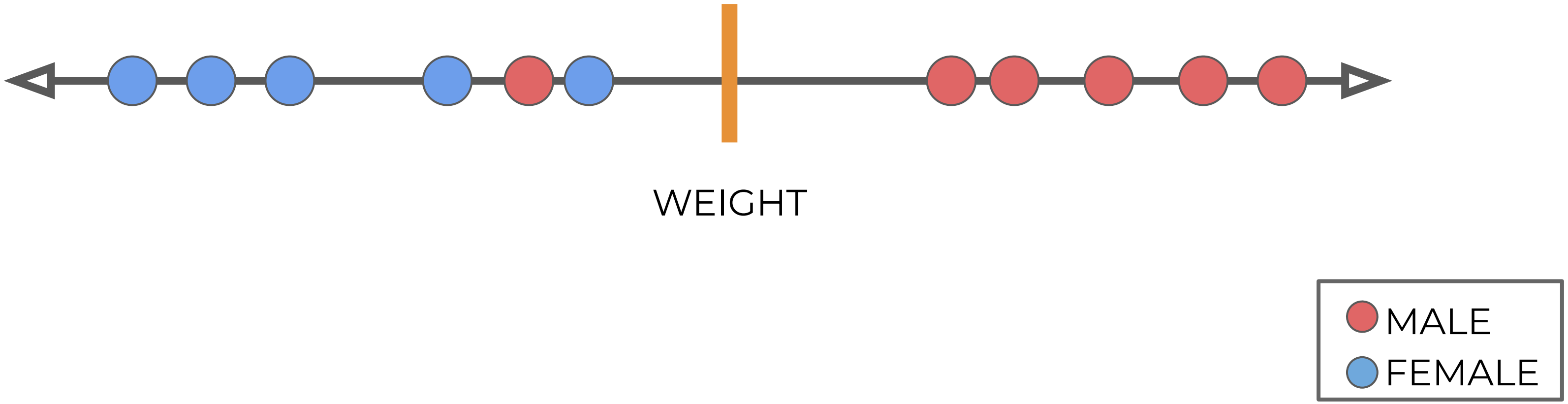
Support Vector Machines

- We are not be able to separate without allowing for misclassifications.



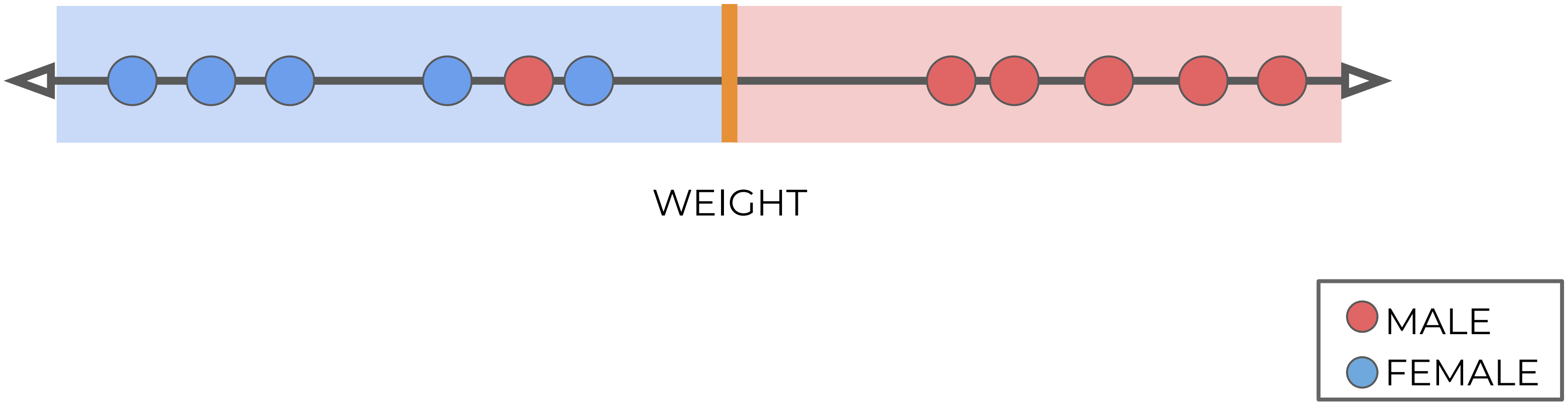
Support Vector Machines

- We will have a bias-variance trade-off depending where we place this separator:



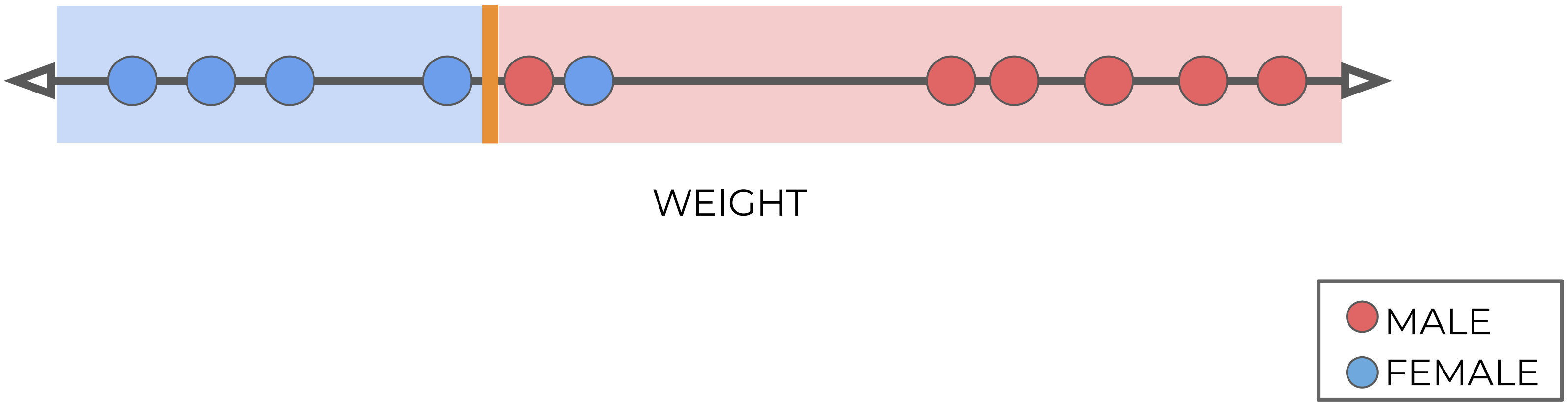
Support Vector Machines

- For one feature this classifier creates ranges for male and female:



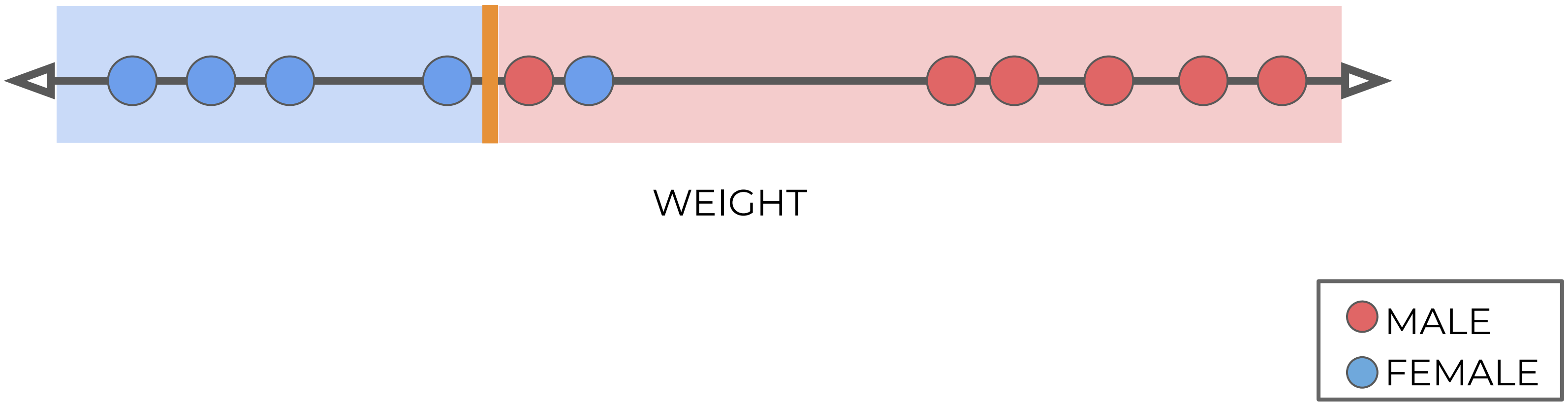
Support Vector Machines

- This fit only misclassified one female training point as male:



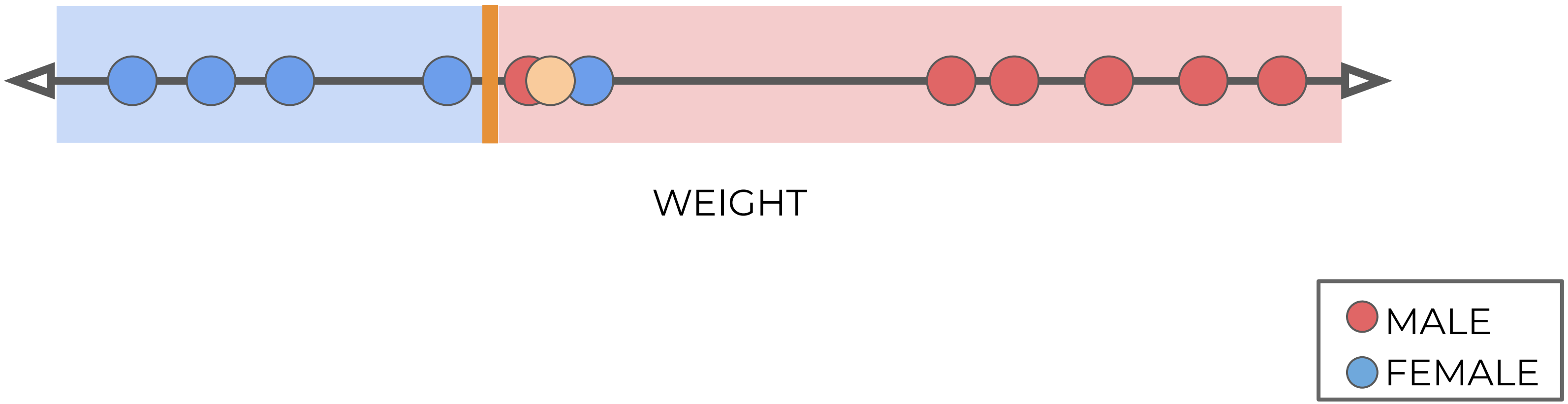
Support Vector Machines

- This looks like a high variance fit to training data, picking too much noise from Female:



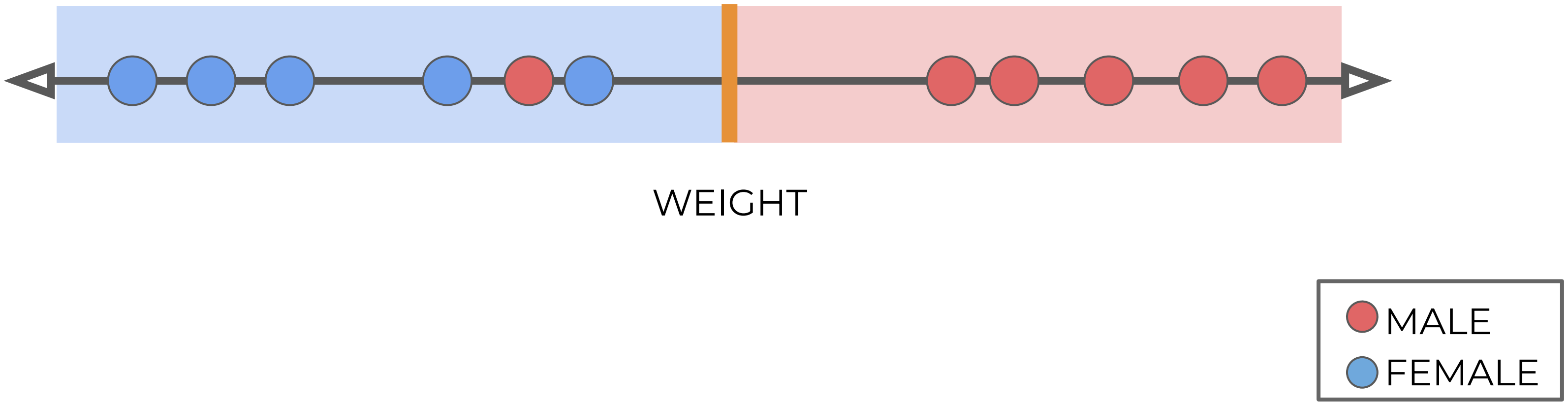
Support Vector Machines

- A new test point close to existing female weights could get classified as male:



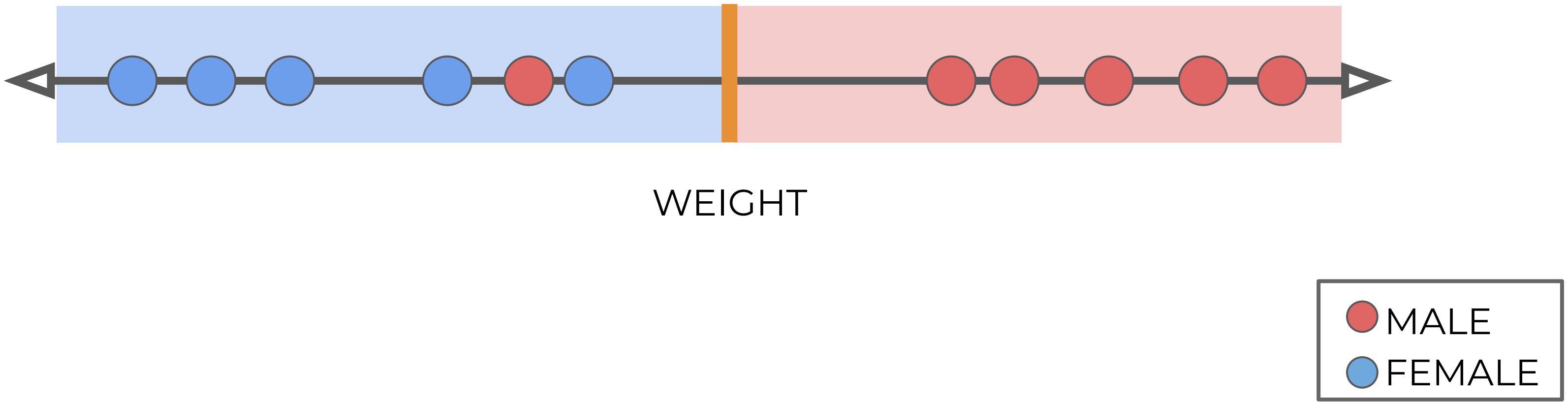
Support Vector Machines

- We will have a bias-variance trade-off depending where we place this separator:



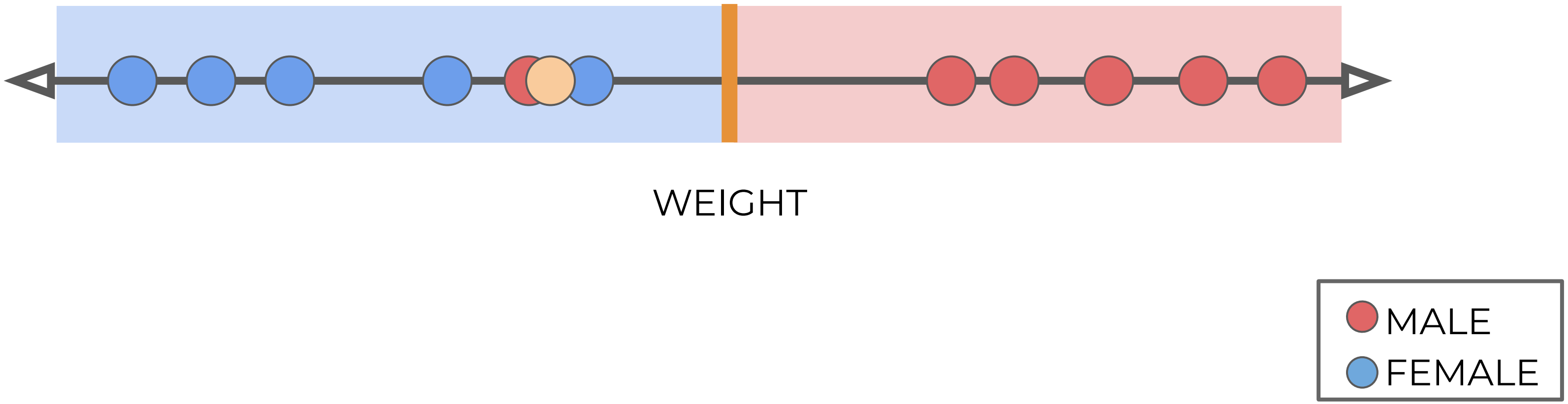
Support Vector Machines

- Here we allow more bias to lead to better long term results on future data:



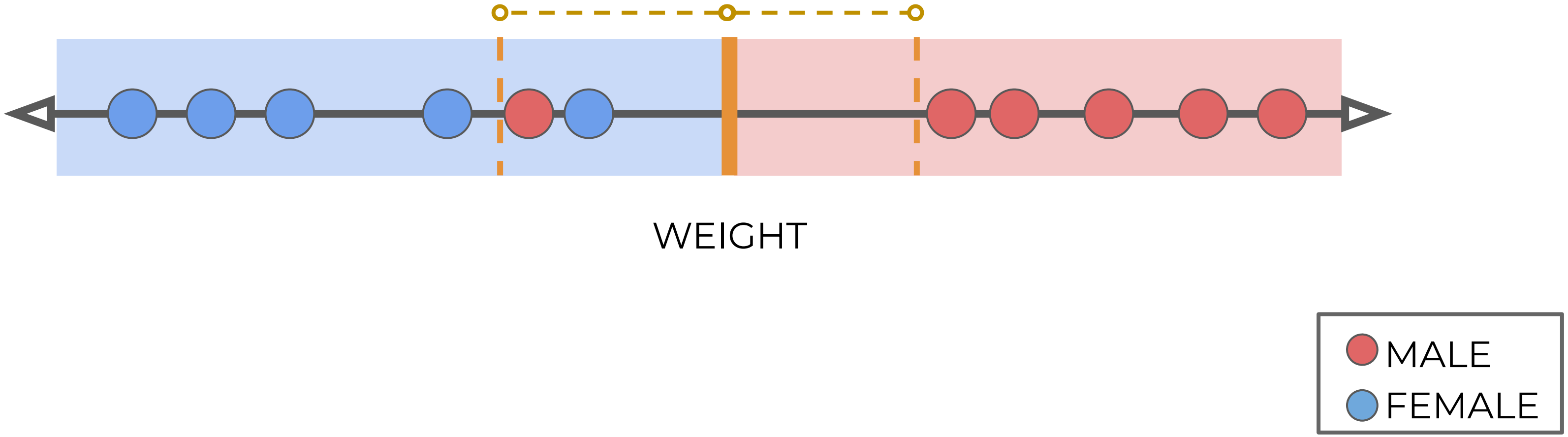
Support Vector Machines

- Here we allow more bias to lead to better long term results on future data:



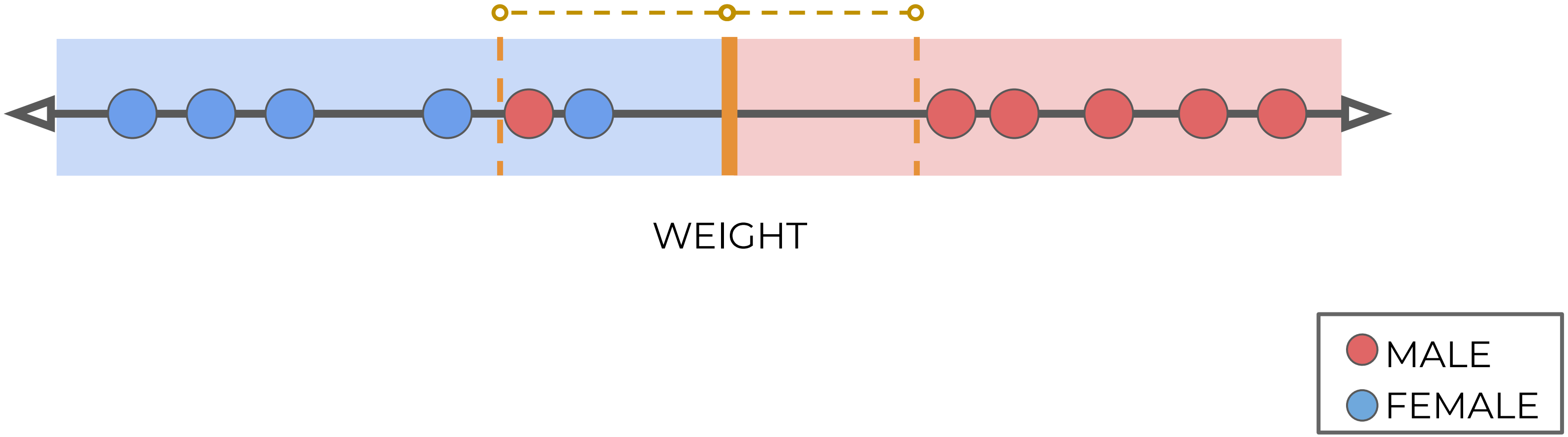
Support Vector Machines

- Distance between threshold and the observations is a **soft margin:**



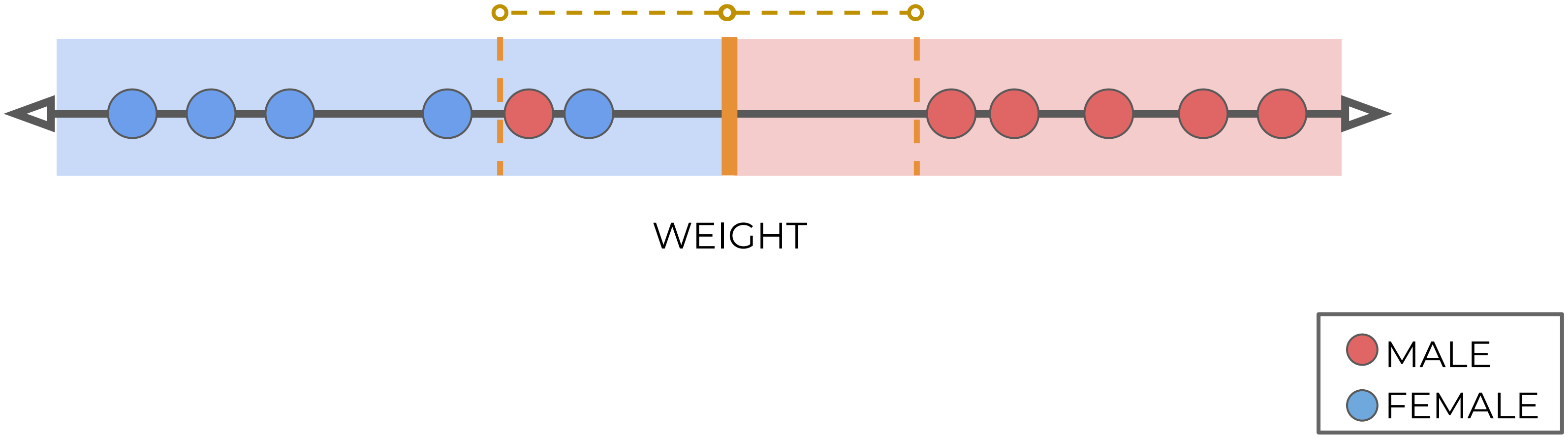
Support Vector Machines

- **Soft margin** allows for misclassification inside the margins.



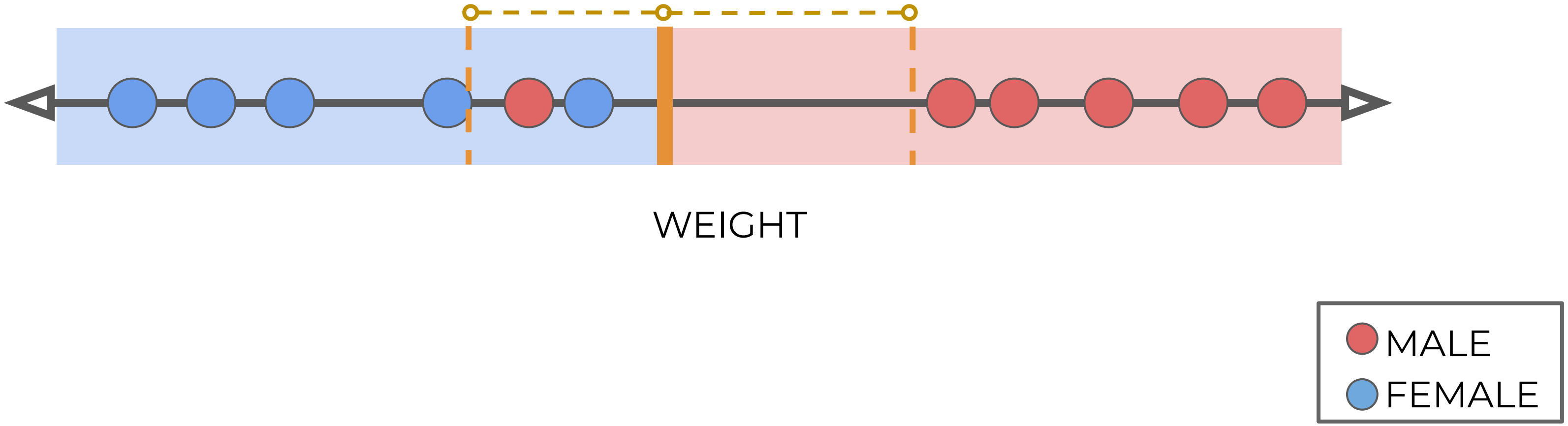
Support Vector Machines

- There are many possible threshold splits if we allow for soft margins.



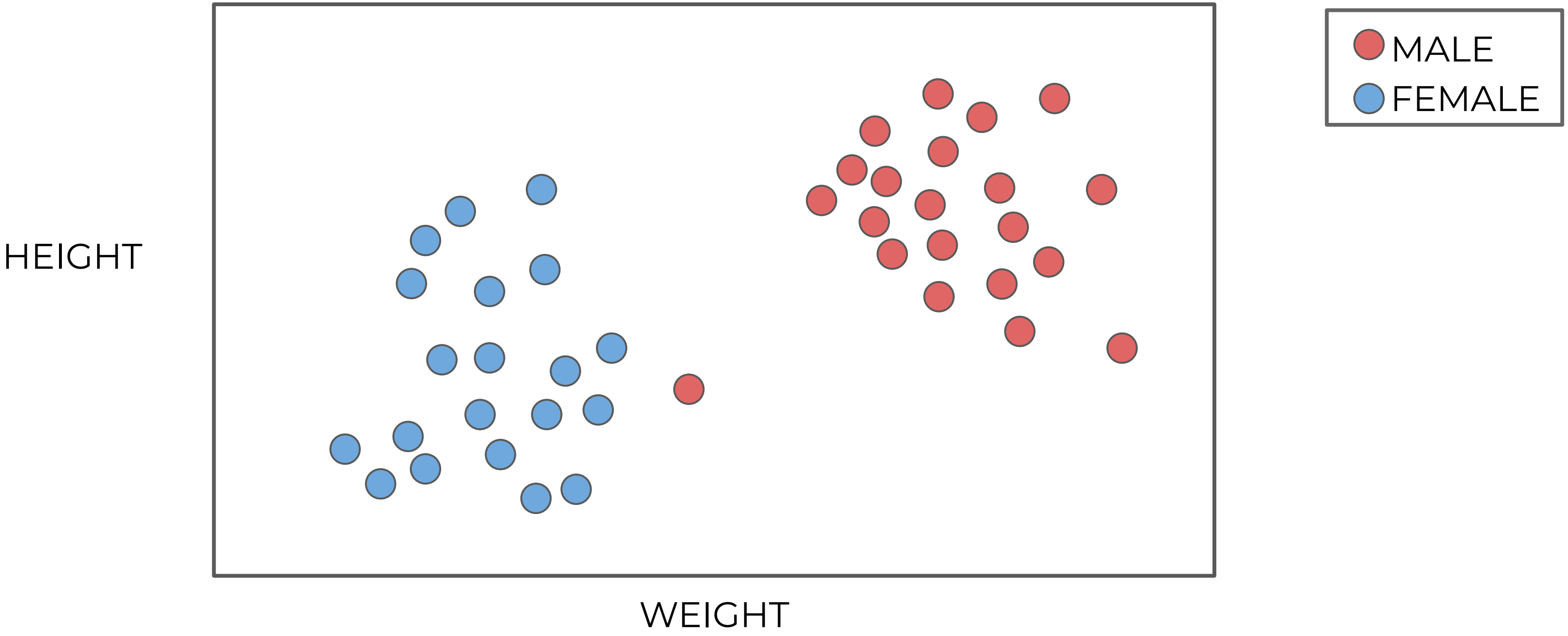
Support Vector Machines

- We can use cross validation to determine the optimal size of the margins.



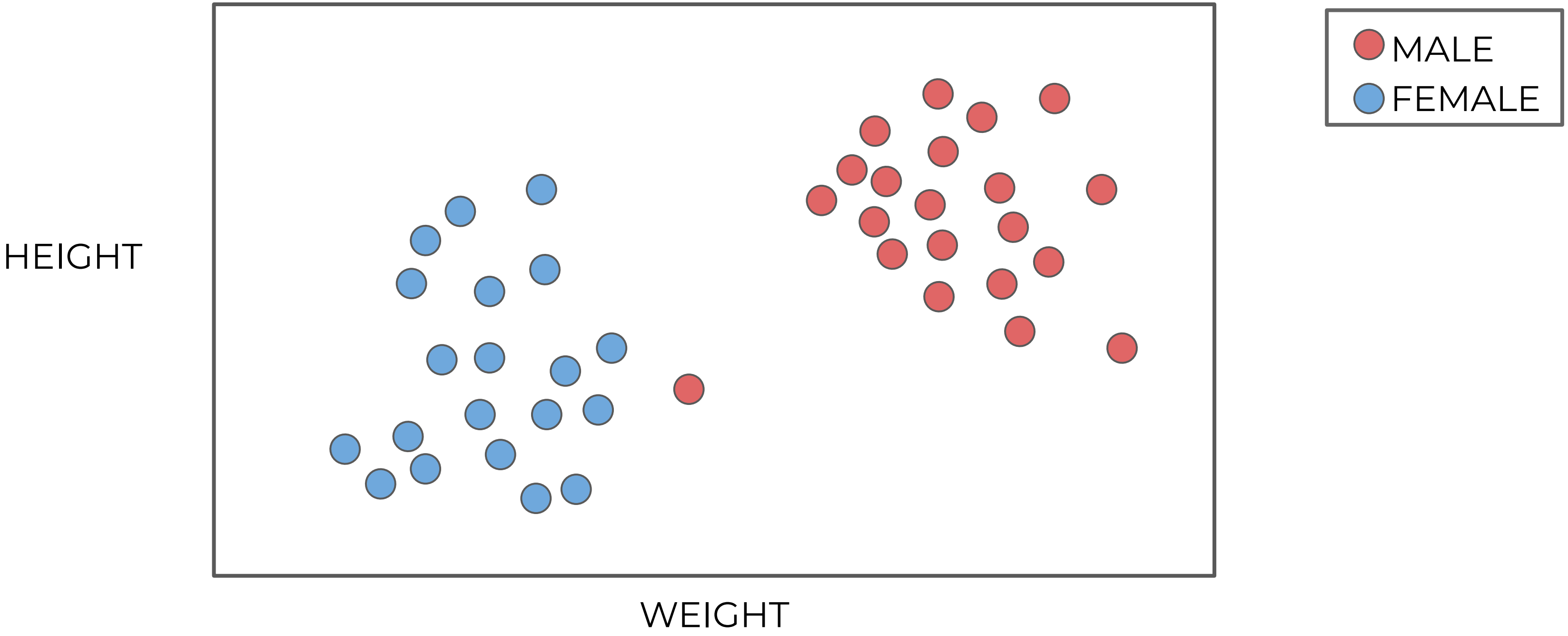
Support Vector Machines

- 2D soft margin example:



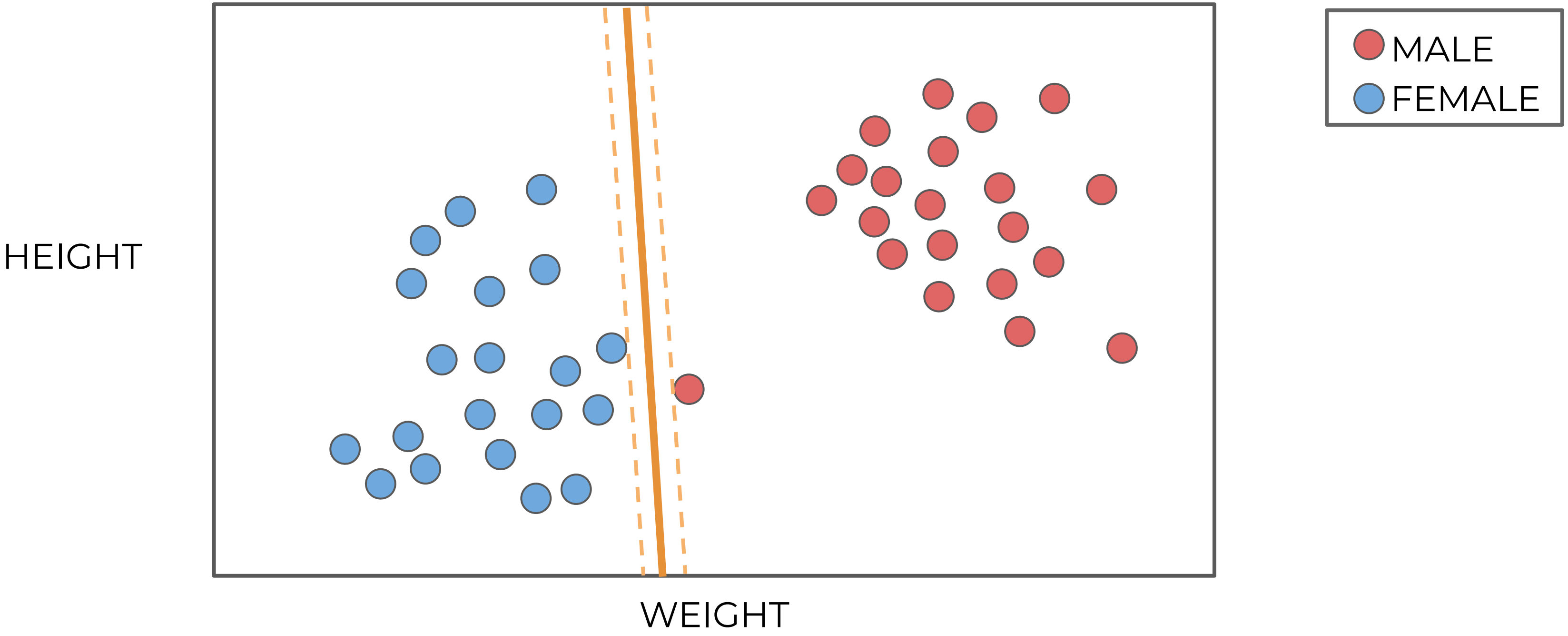
Support Vector Machines

- Data set is technically perfectly separable



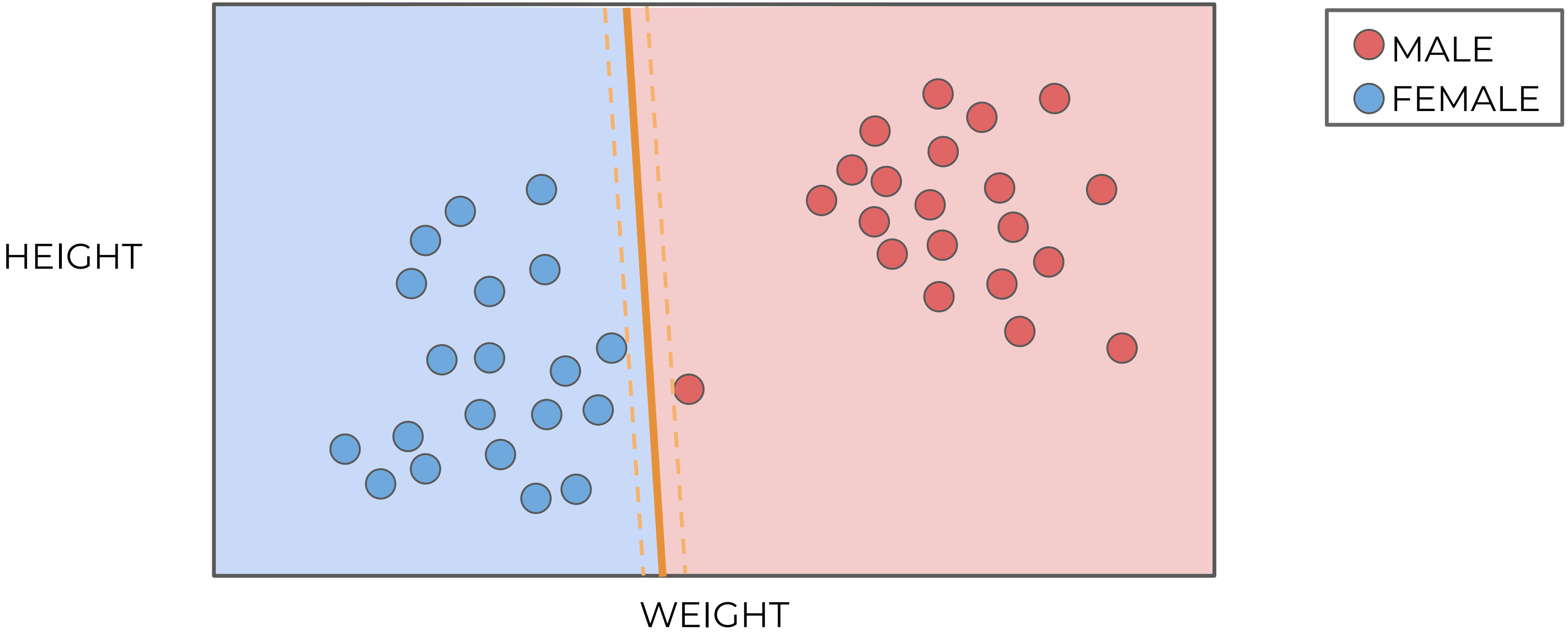
Support Vector Machines

- Maximal Margin Classifier



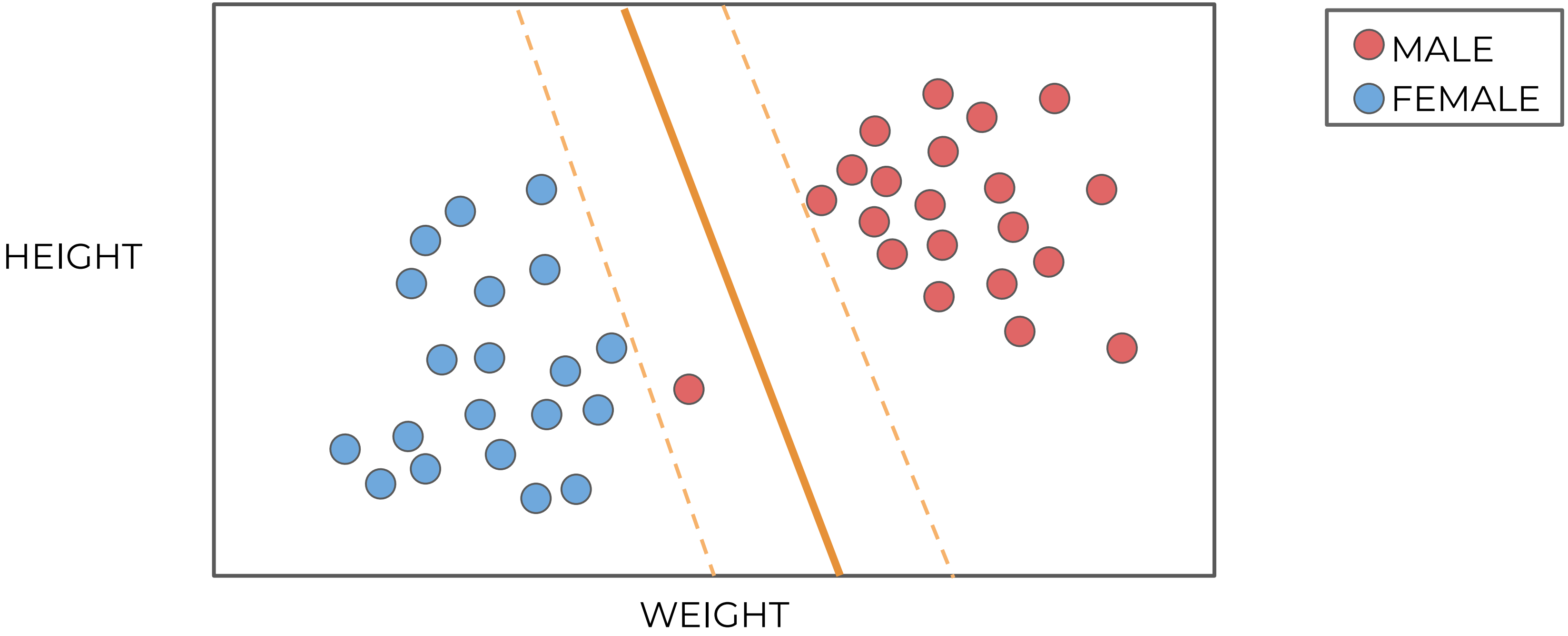
Support Vector Machines

- Maximal Margin Classifier



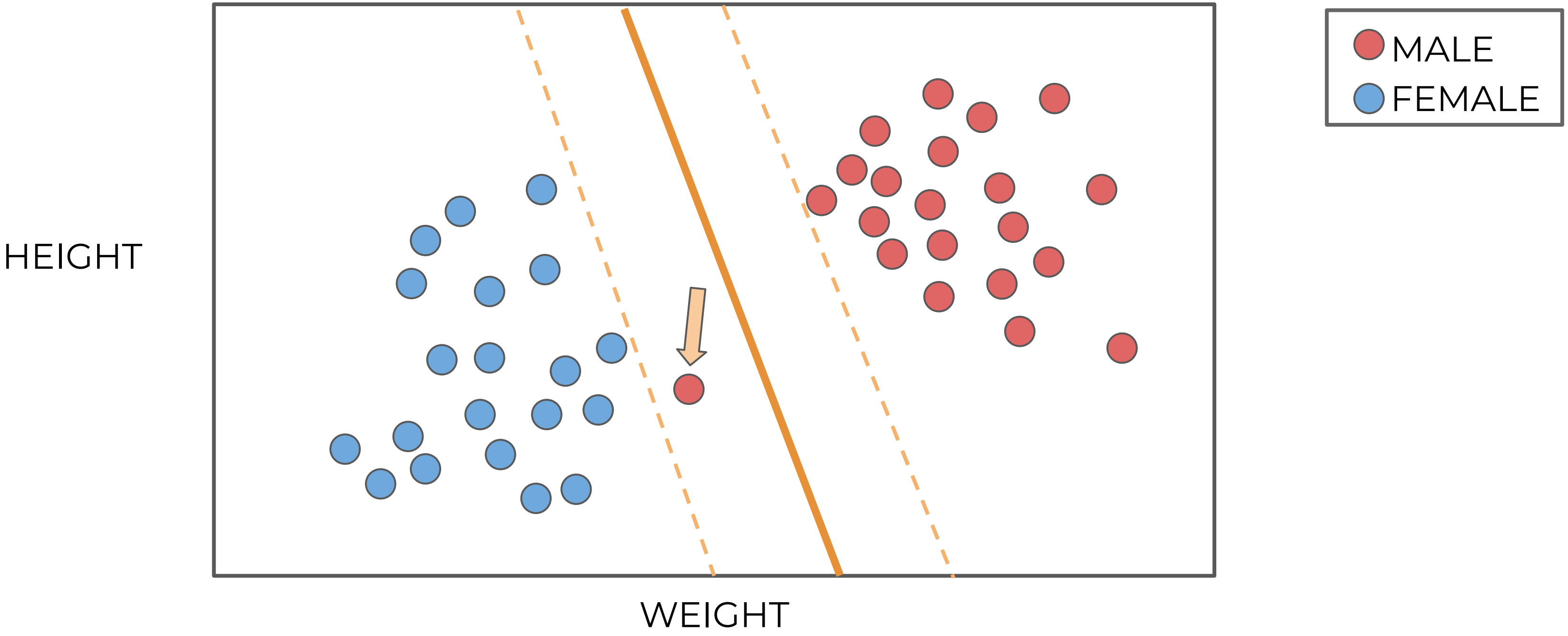
Support Vector Machines

- Support Vector Classifier (Soft Margins)



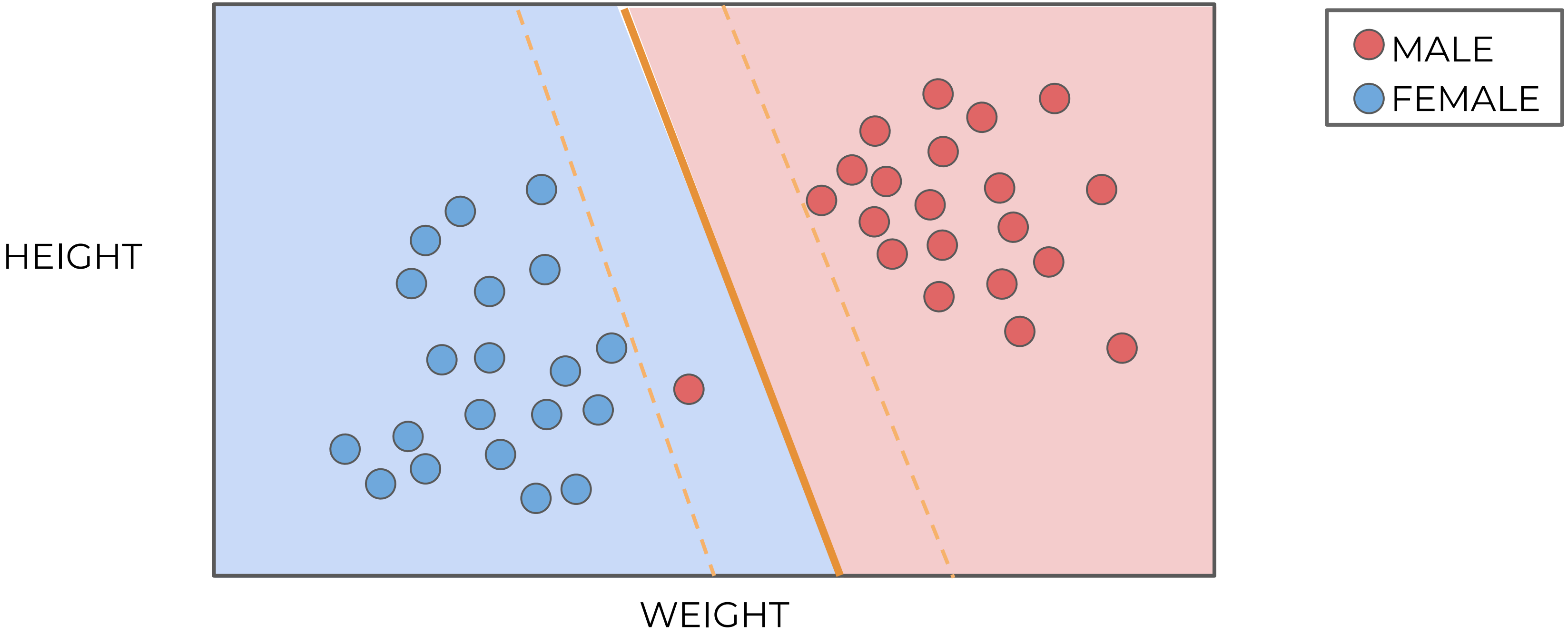
Support Vector Machines

- Support Vector Classifier (Soft Margins)



Support Vector Machines

- Support Vector Classifier (Soft Margins)

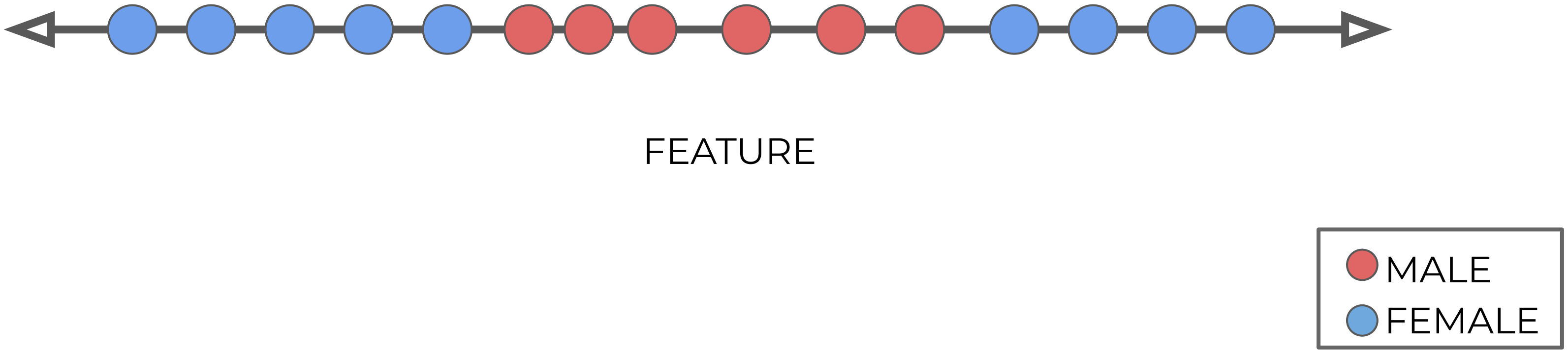


Support Vector Machines

- We’ve only visualized cases where the classes are easily separated by the hyperplane in the original feature space.
- Allowing for some misclassifications still resulted in reasonable results.
- What would happen in a case where a hyperplane performs poorly, even when allowing for misclassifications?

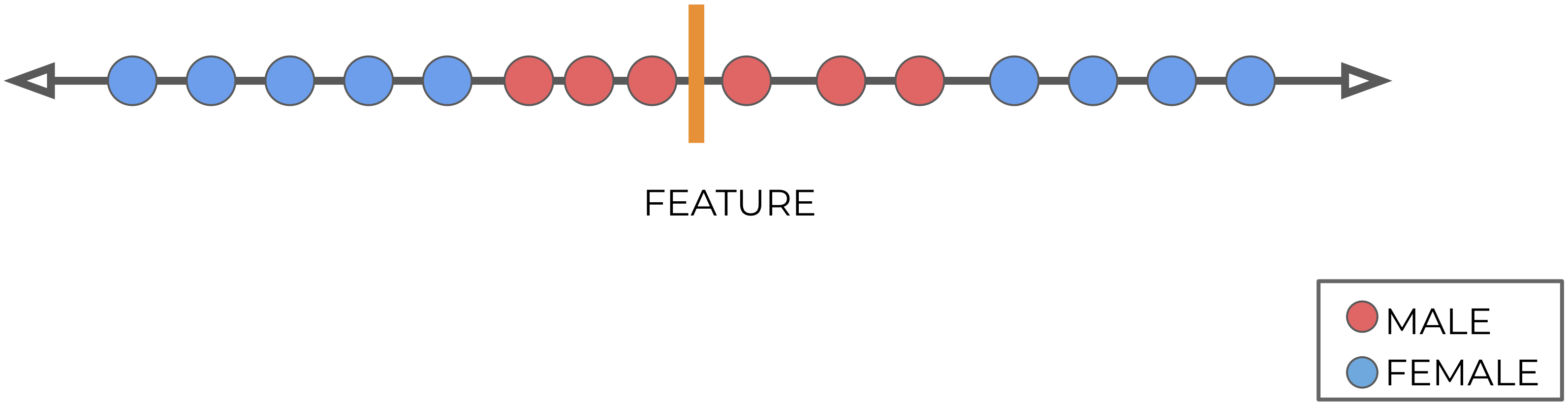
Support Vector Machines

- Notice a single hyperplane won't separate out the classes without many misclassifications!



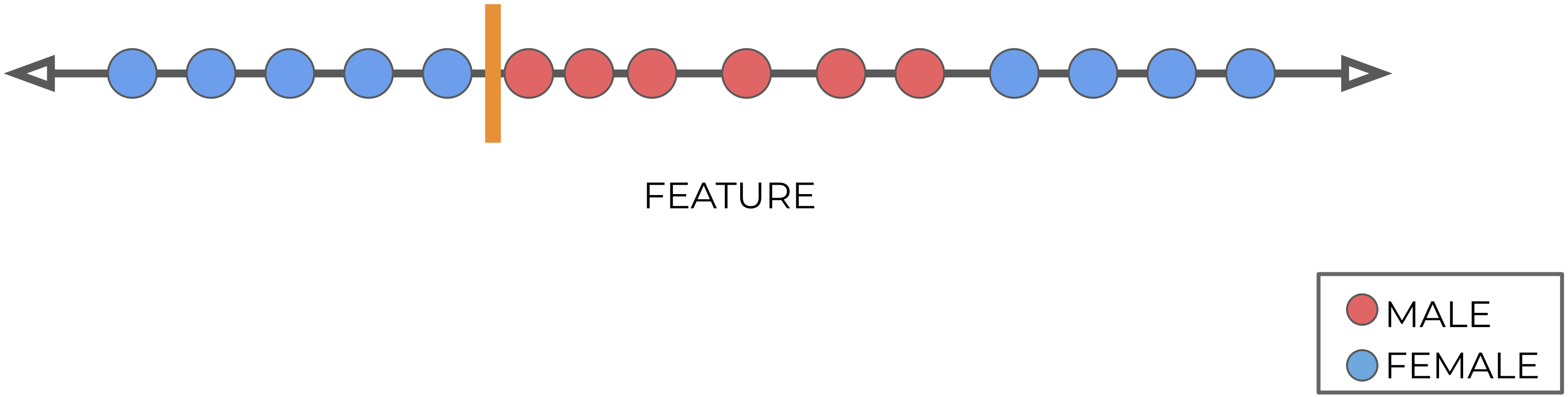
Support Vector Machines

- Notice a single hyperplane won't separate out the classes without many misclassifications!



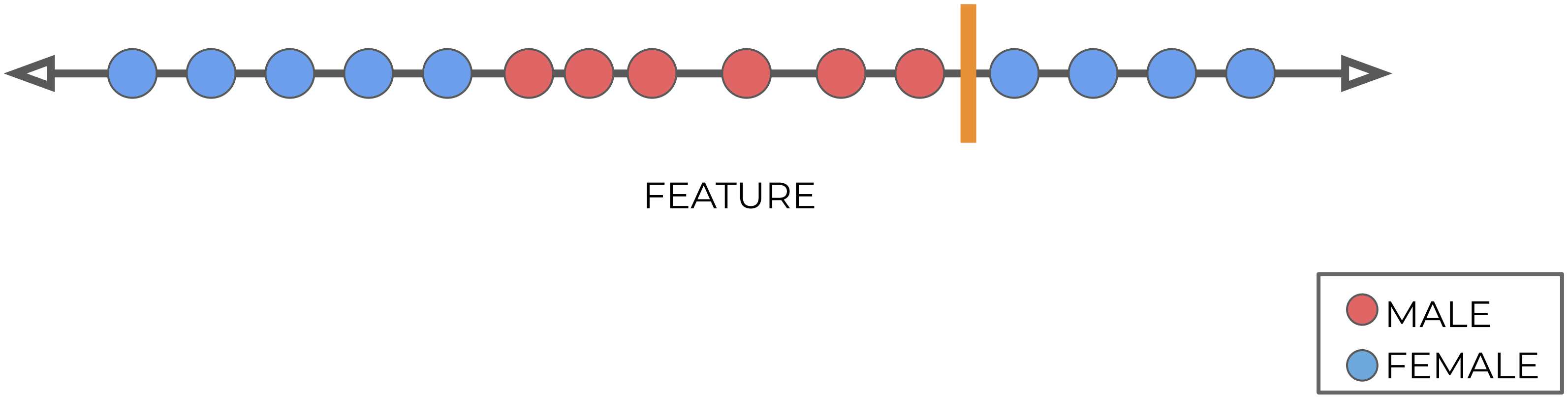
Support Vector Machines

- Notice a single hyperplane won't separate out the classes without many misclassifications!



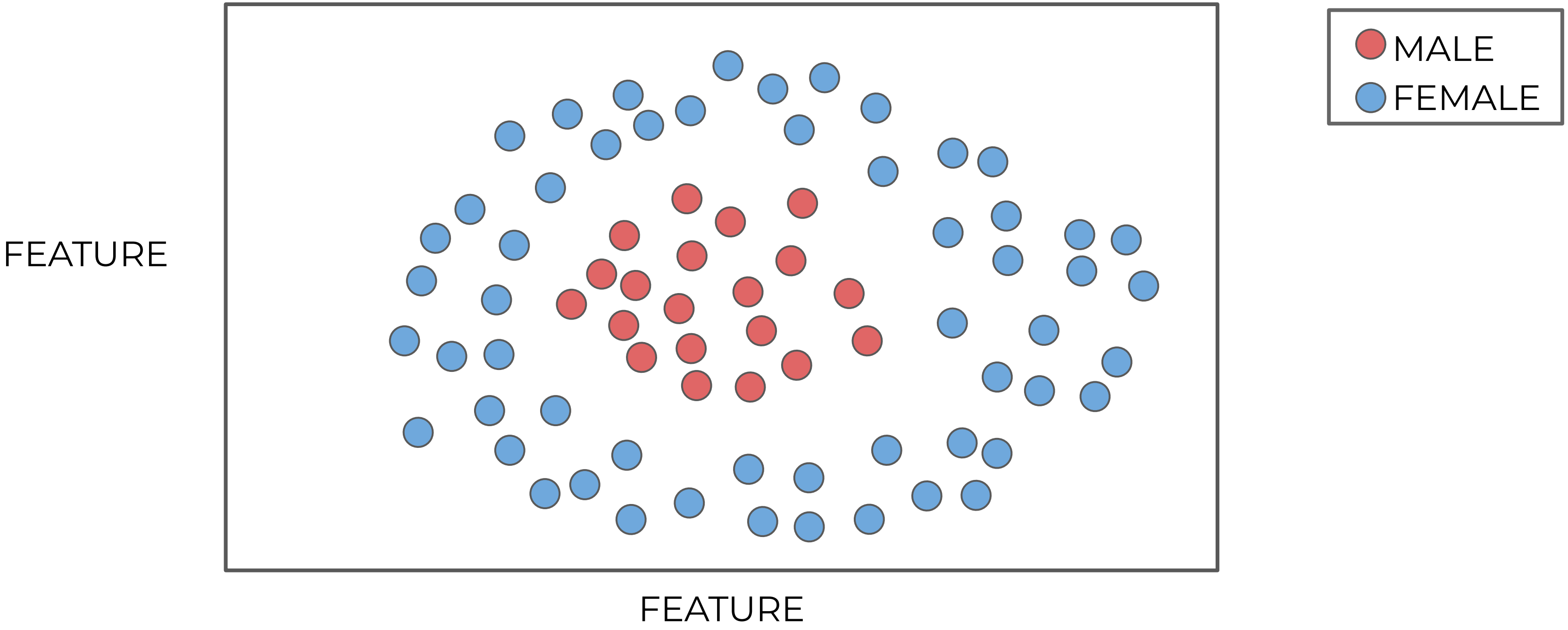
Support Vector Machines

- Notice a single hyperplane won't separate out the classes without many misclassifications!



Support Vector Machines

- Can’t split classes with hyperplane line:



Support Vector Machines

- To solve these cases, we move on from Support Vector Classifier, to Support Vector Machines.
- SVMs use **kernels** to project the data to a higher dimension, in order to use a hyperplane in this higher dimension to separate the data.

Support Vector Machines

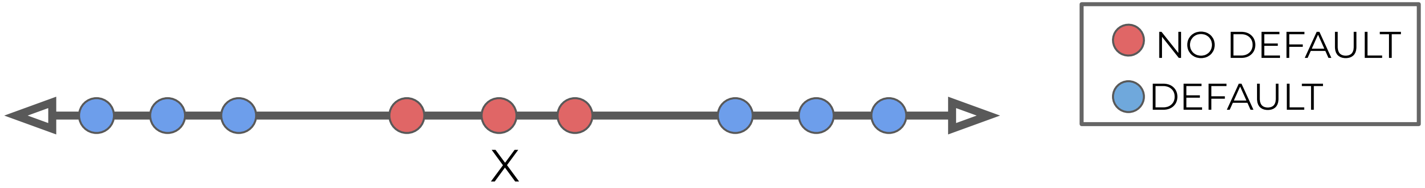
Theory and Intuition - Kernels

Support Vector Machines

- Kernels allow us to move beyond a Support Vector Classifier and use Support Vector Machines.
- There are a variety of kernels we can use to “project” the features to a higher dimension.
- Let’s explore how this works through some visual examples…

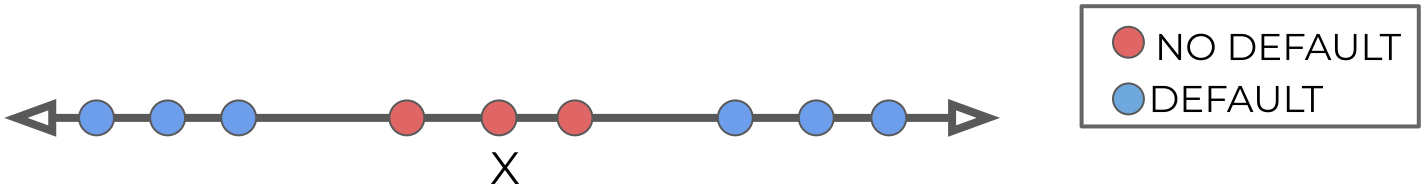
Support Vector Machines

- Recall our 1D example of classes not easily separated by a single hyperplane:



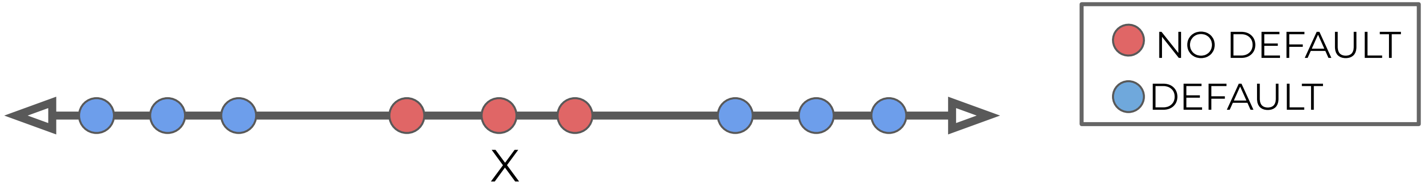
Support Vector Machines

- Let’s explore how using a kernel could project this feature onto another dimension.



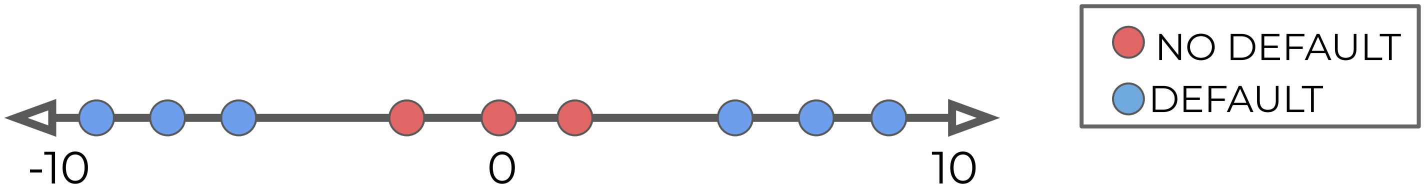
Support Vector Machines

- For example, a polynomial kernel could expand onto an X^2 dimension:



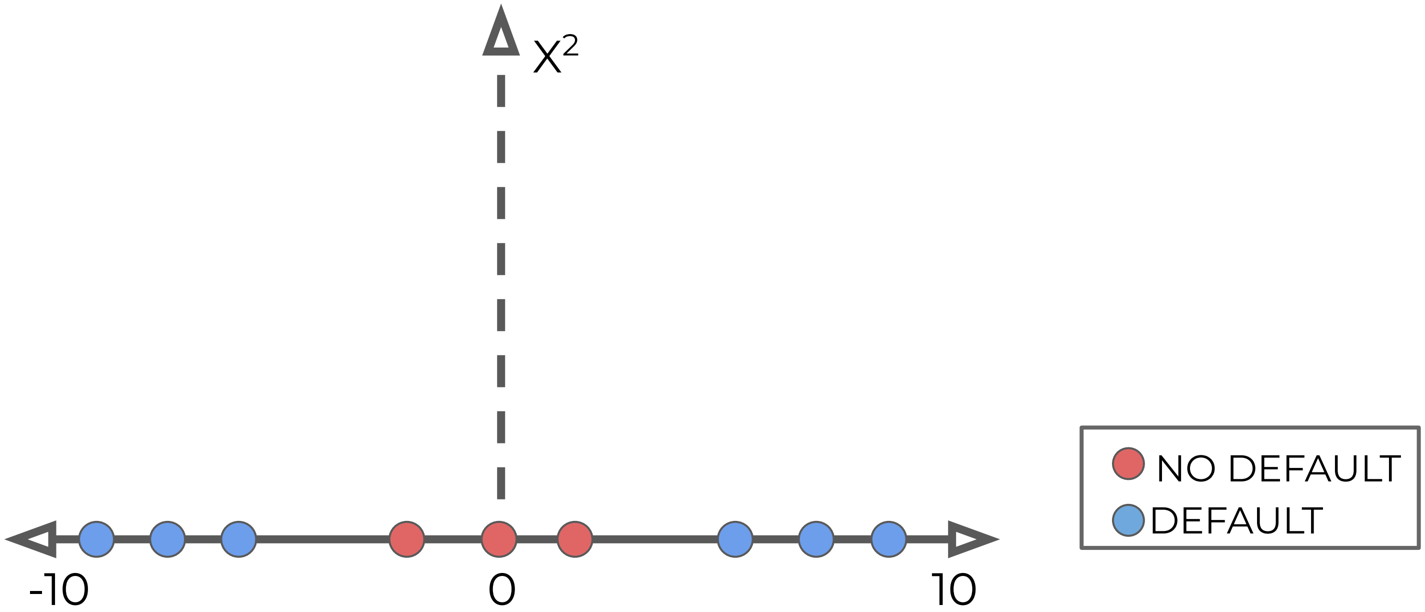
Support Vector Machines

- For example, a polynomial kernel could expand onto an X^2 dimension:



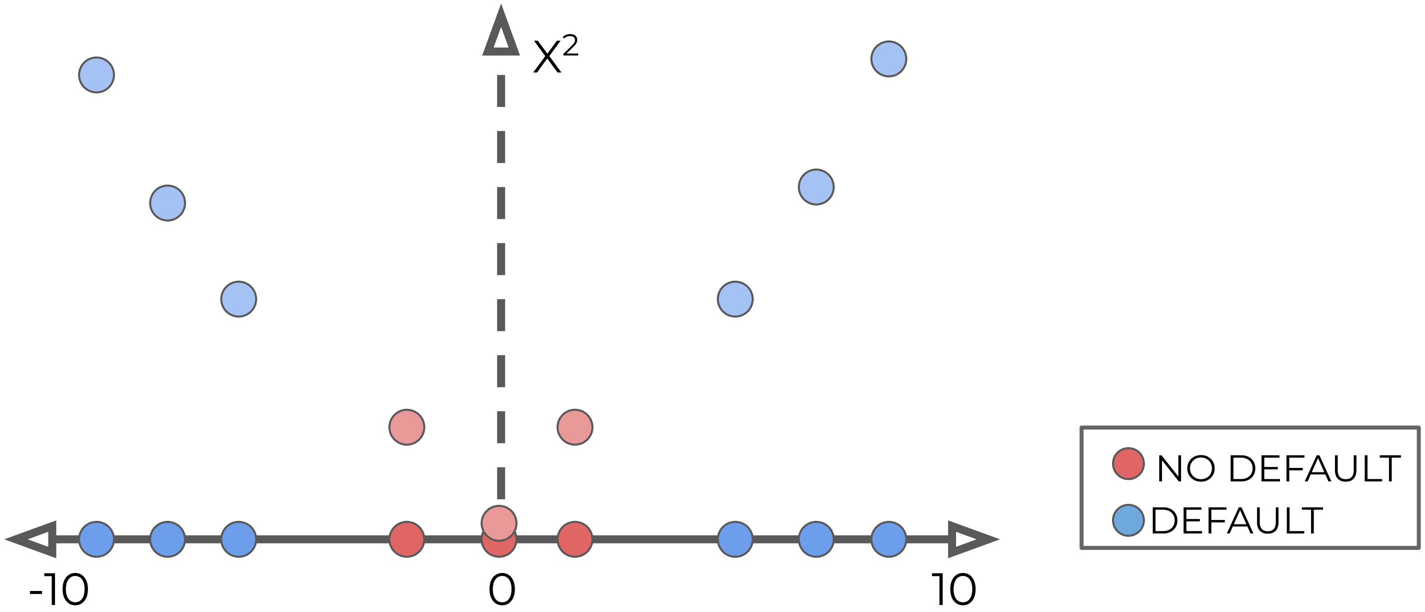
Support Vector Machines

- For example, a polynomial kernel could expand onto an X^2 dimension:



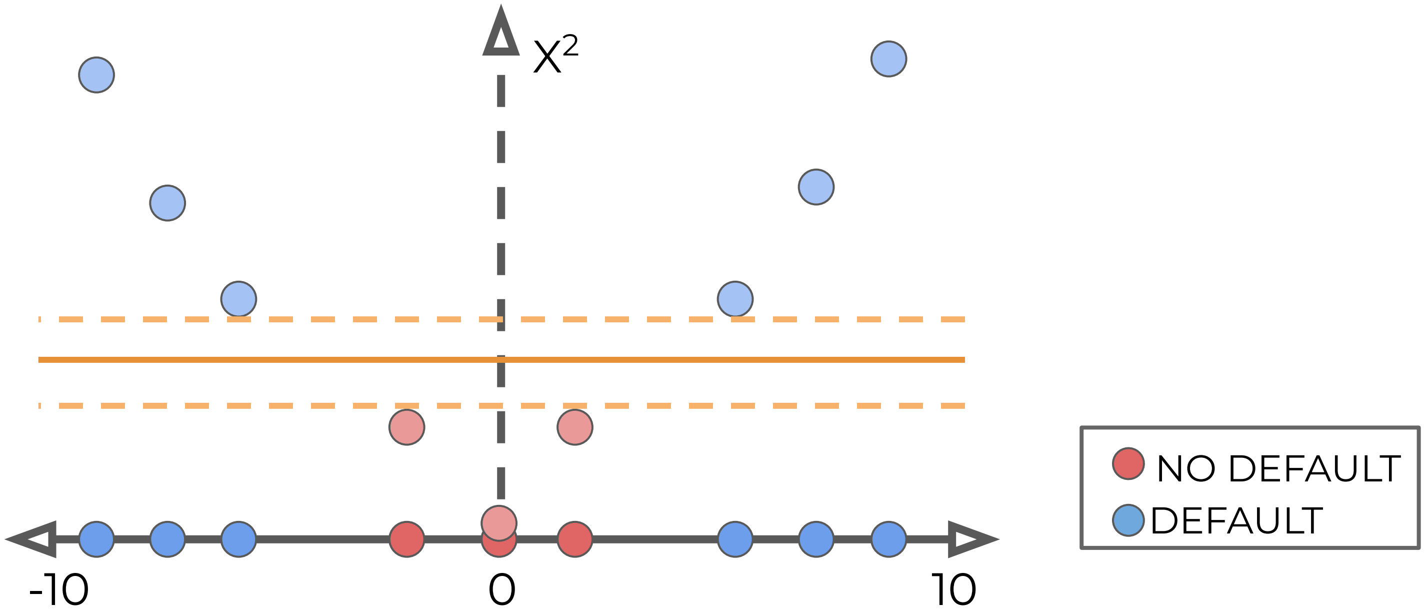
Support Vector Machines

- For example, a polynomial kernel could expand onto an X^2 dimension:



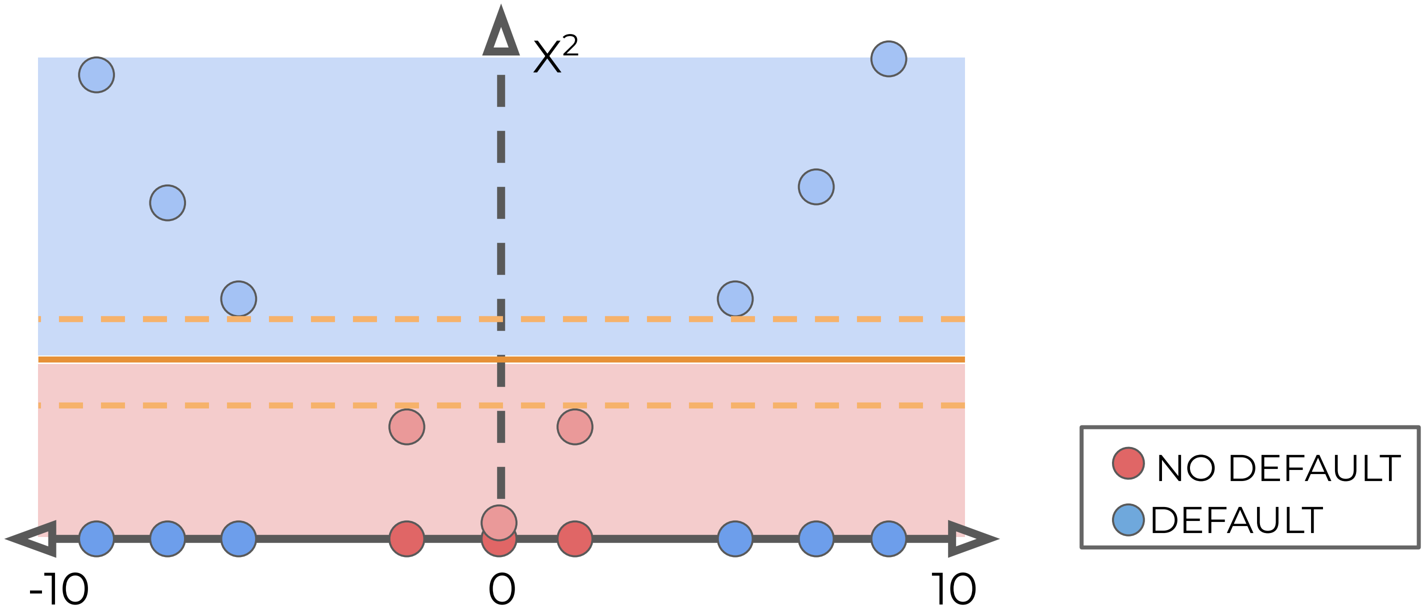
Support Vector Machines

- Create a hyperplane after this projection:



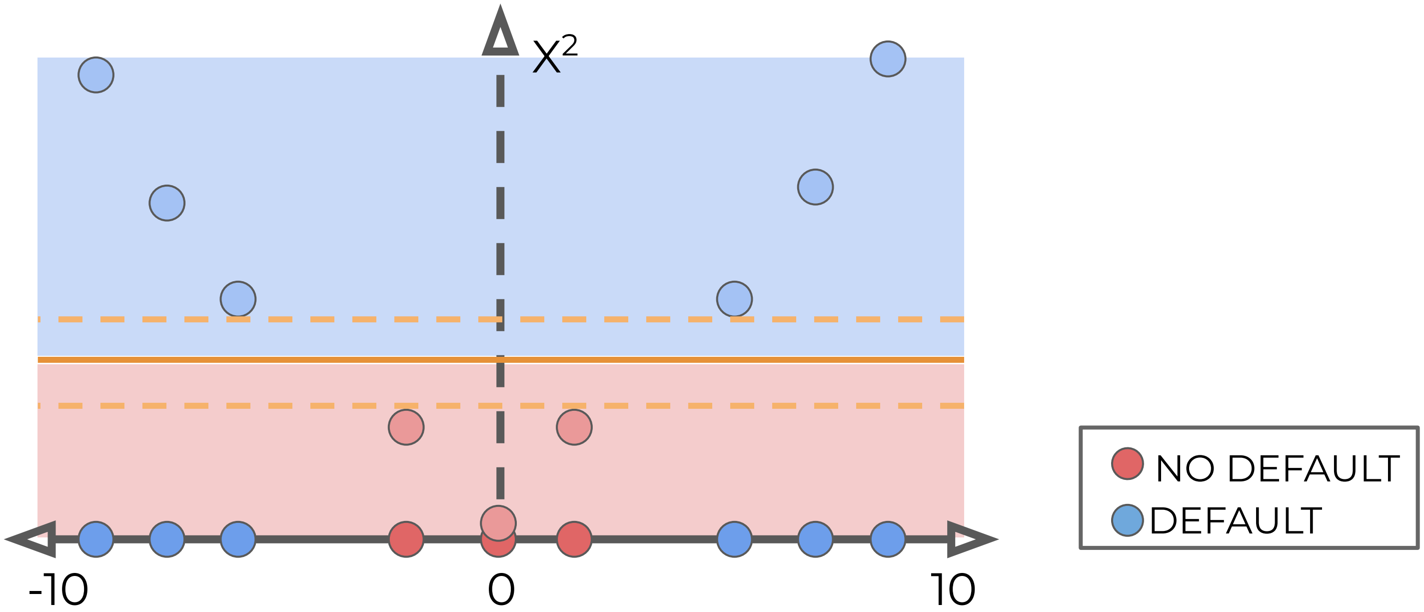
Support Vector Machines

- Create a hyperplane after this projection:



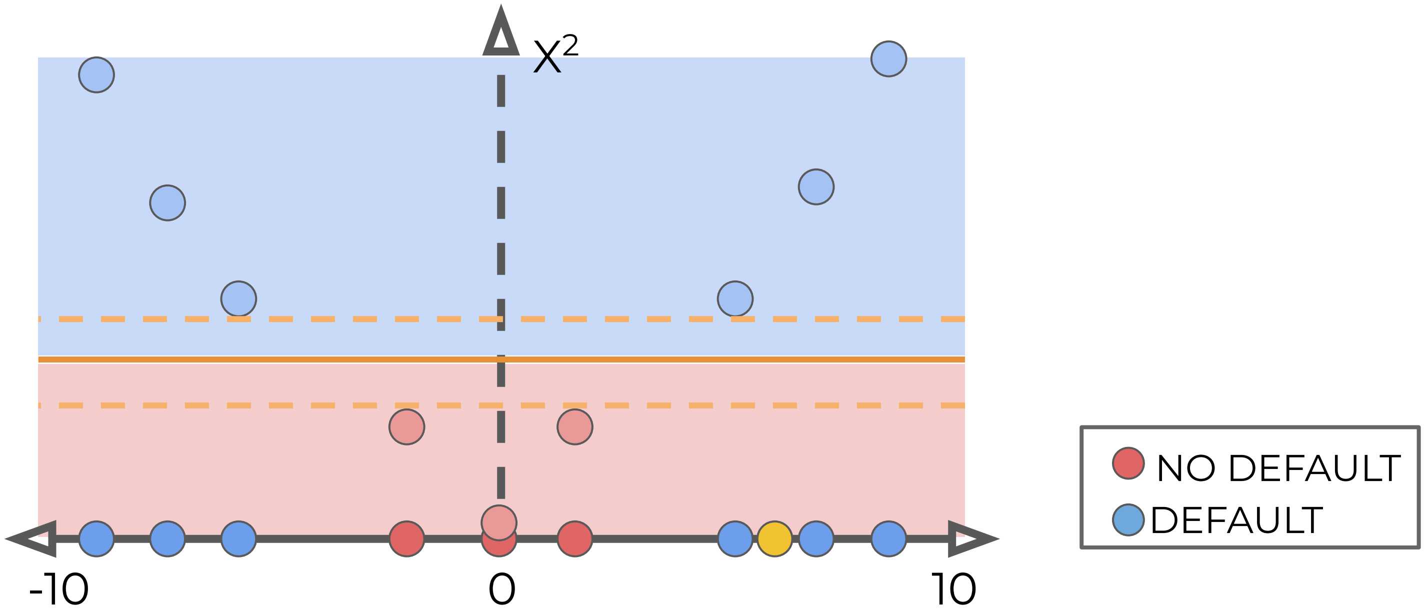
Support Vector Machines

- Use kernel projection to evaluate new points:



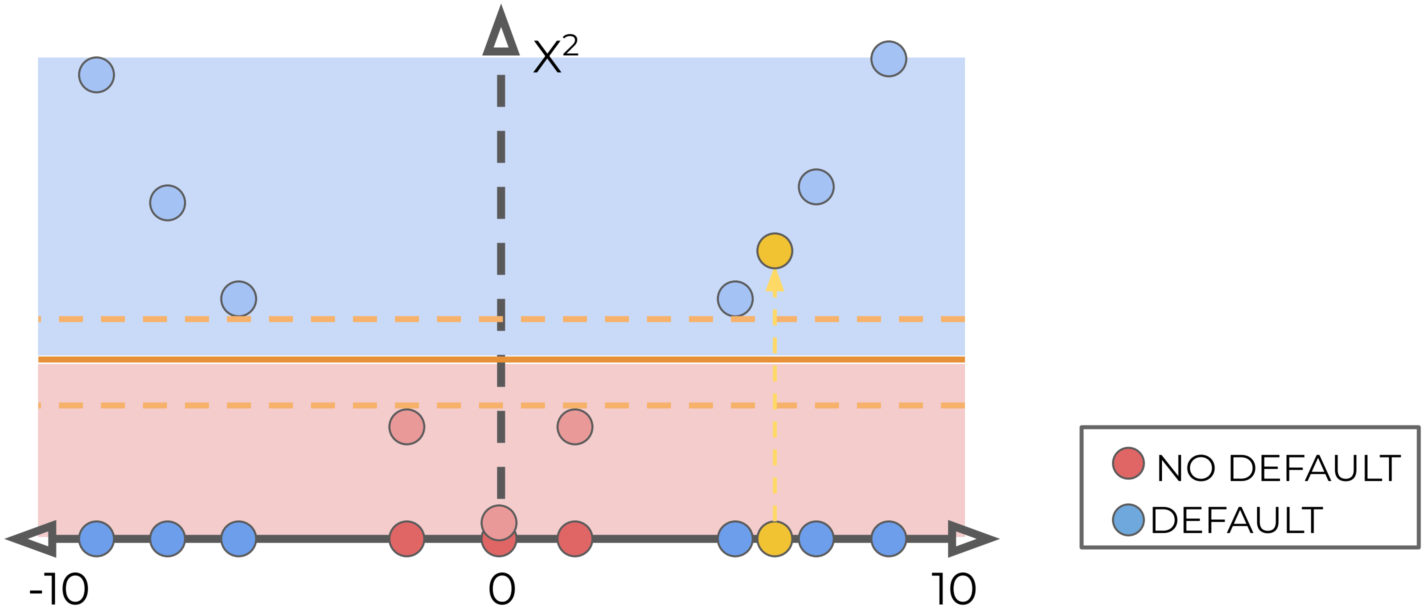
Support Vector Machines

- Use kernel projection to evaluate new points:



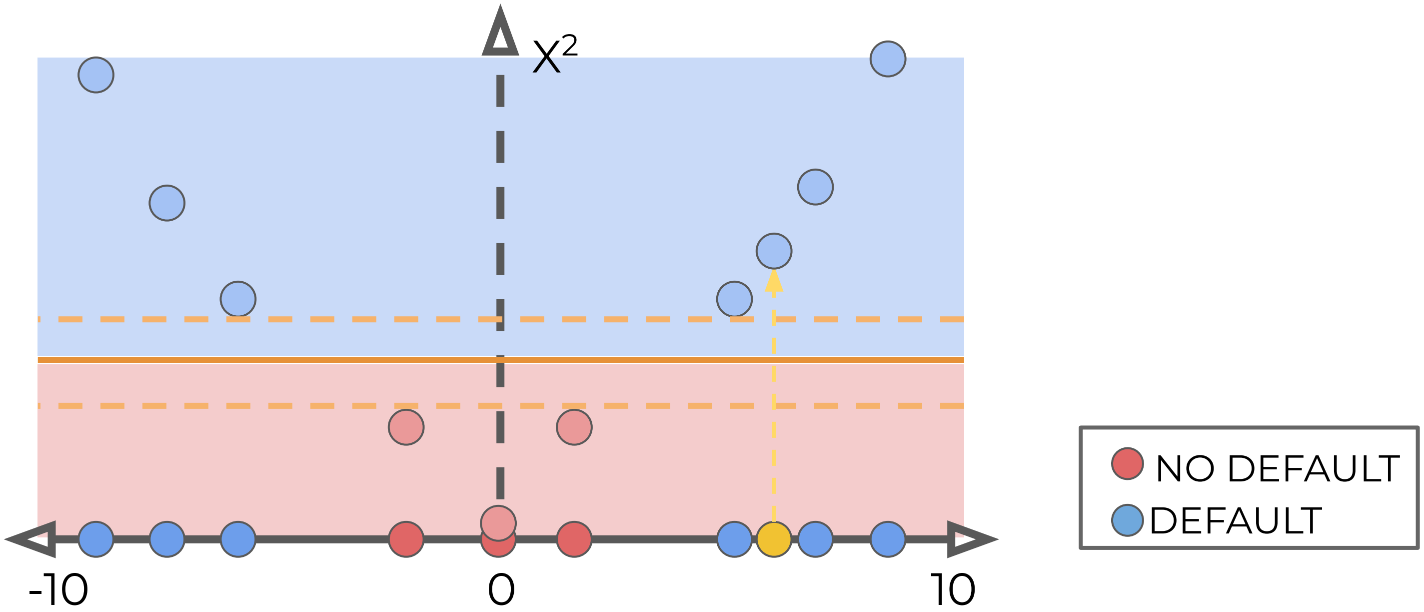
Support Vector Machines

- Use kernel projection to evaluate new points:



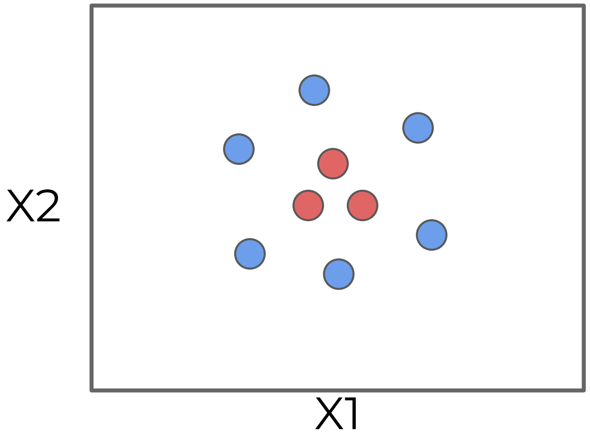
Support Vector Machines

- Use kernel projection to evaluate new points:



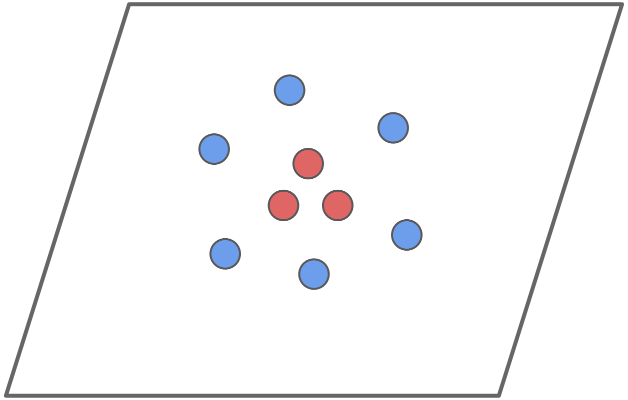
Support Vector Machines

- Imagine a 2D feature space where a hyperplane can not separate effectively, even with soft margins.



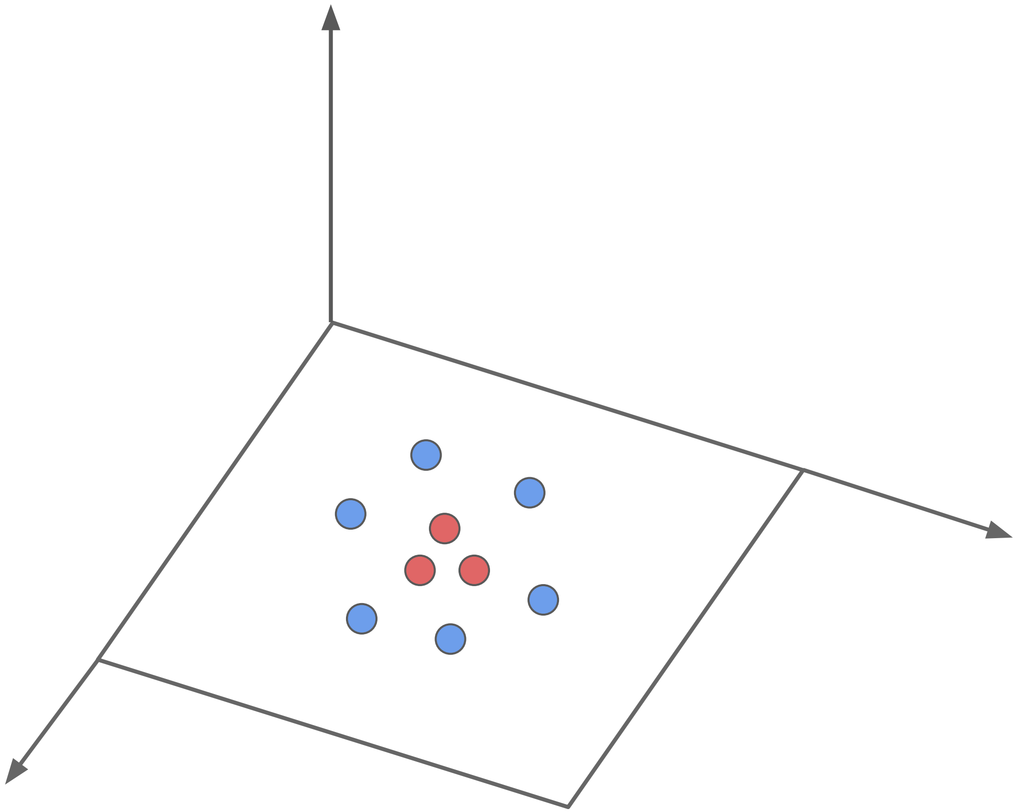
Support Vector Machines

- We use Support Vector Machines to enable the use of a kernel transformation to project to a higher dimension.



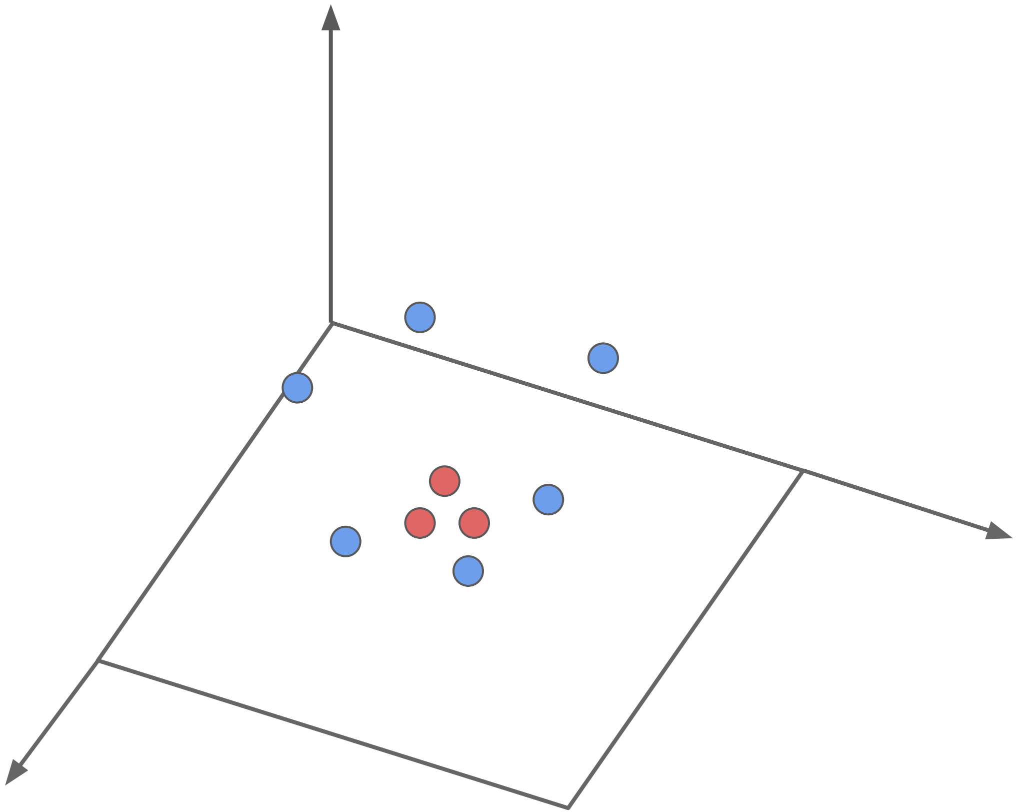
Support Vector Machines

- 2D to 3D



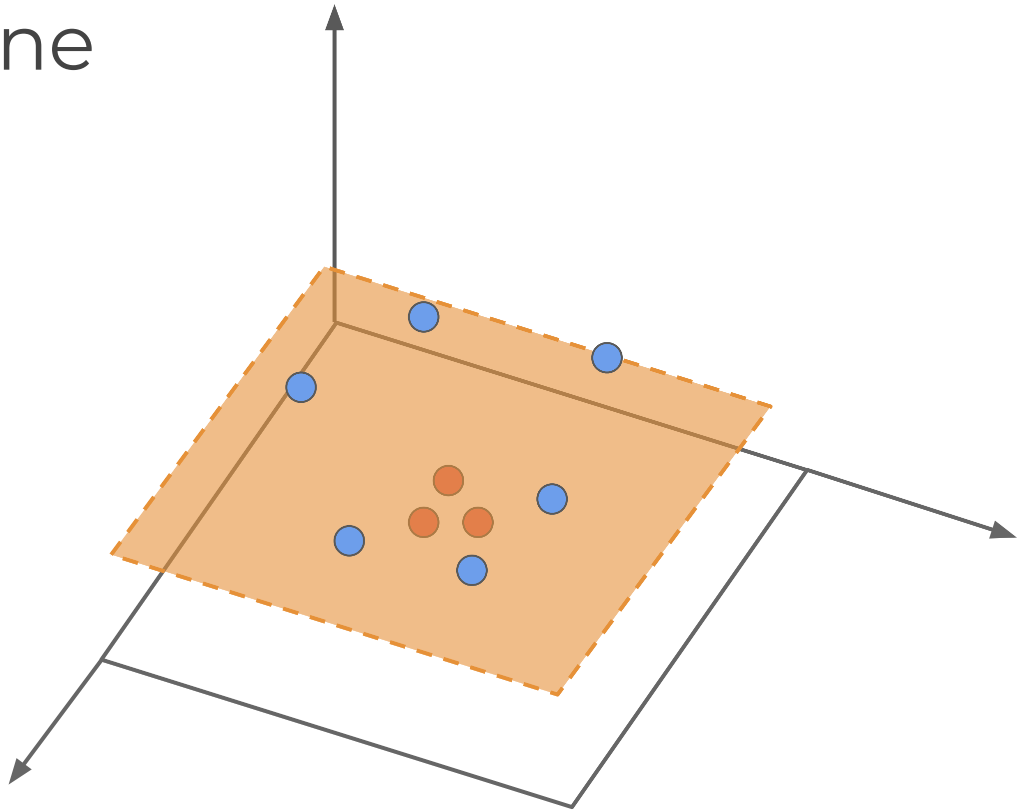
Support Vector Machines

- 2D to 3D



Support Vector Machines

- Hyperplane



Support Vector Machines

- You may have heard of the use of kernels in SVM as the **"kernel trick"**.
- We previously visualized transforming data points from one dimension into a higher dimension.
- Mathematically, the **kernel trick** actually avoids recomputing the points in a higher dimensional space!

Support Vector Machines

- How does the kernel trick achieve this?
- It takes advantage of dot products of the transpositions of the data.

Support Vector Machines

Theory and Intuition - Kernel Trick and Math

Support Vector Machines

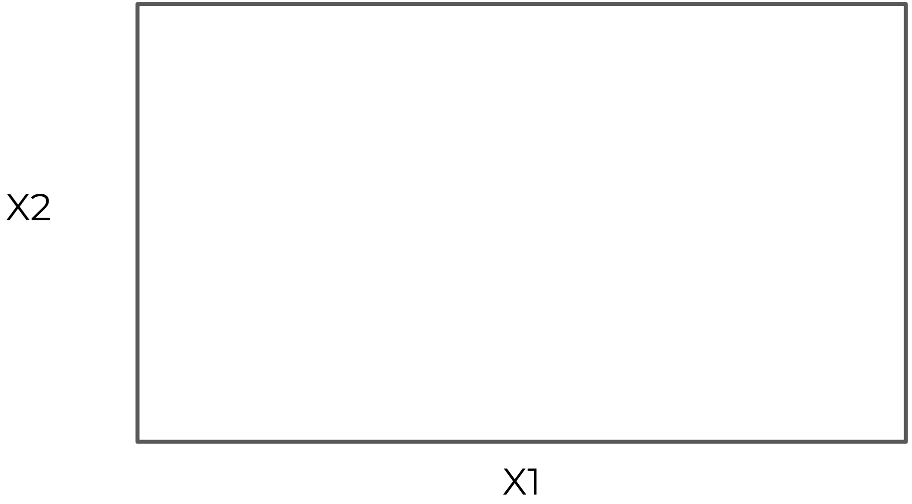
- Let’s briefly go over some of the general mathematics of SVM and how it is related to the Scikit-Learn class calls.
- We’ll begin with a brief review of using margin based classifiers, and how they can be described with equations.
    - *Note: Feel free to consider this an “optional” concept.*

Support Vector Machines

- Relative Reading:
  - Background in Chapter 9 of ISLR.
  - For a comprehensive overview of everything discussed, check out:
    - *Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning.*

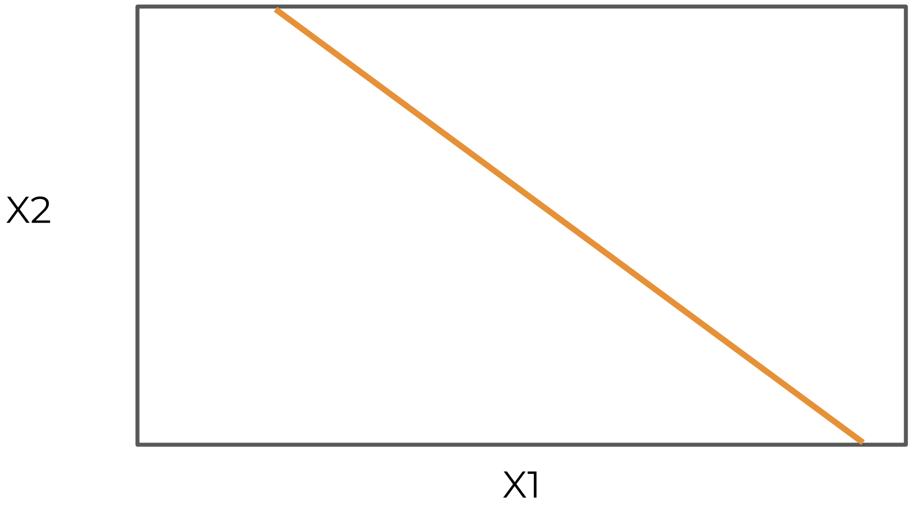
Support Vector Machines

- Hyperplanes Defined



Support Vector Machines

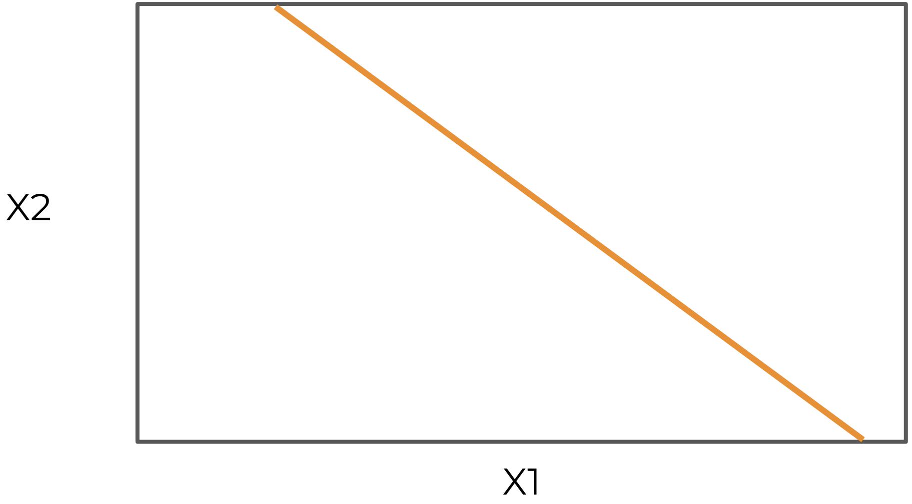
- Hyperplanes Defined



Support Vector Machines

- Hyperplanes Defined

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$



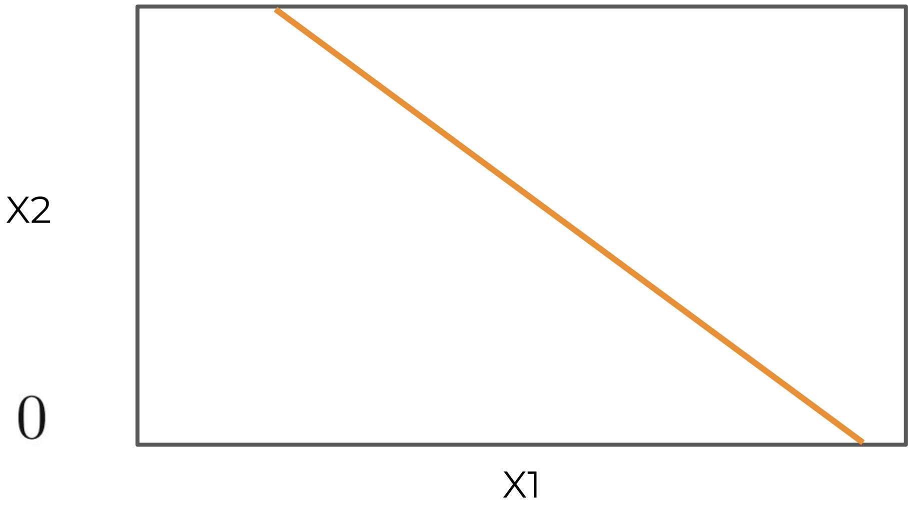
Support Vector Machines

- Hyperplanes Defined

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

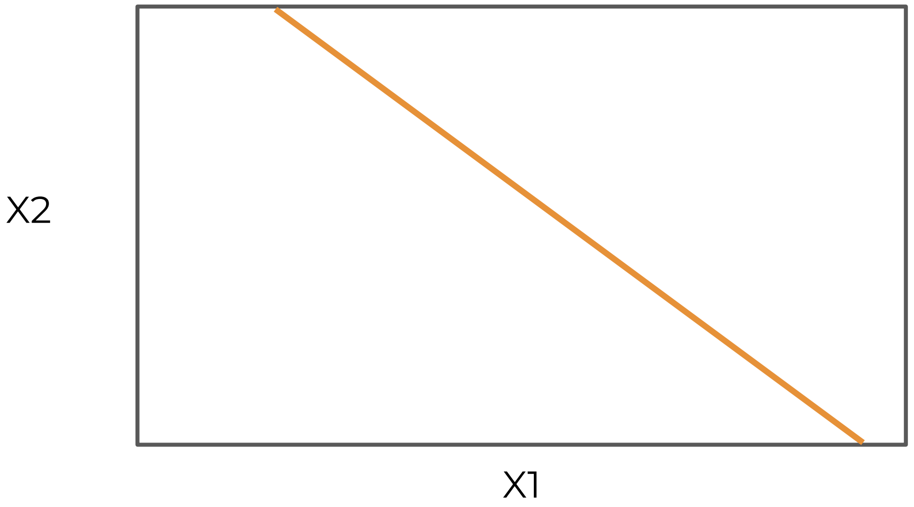
$$\Downarrow$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$



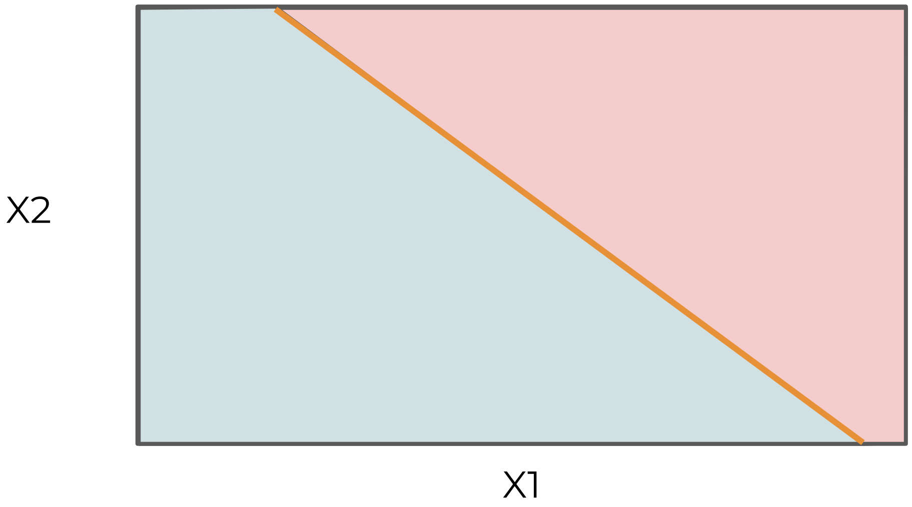
Support Vector Machines

- Separating Hyperplanes



Support Vector Machines

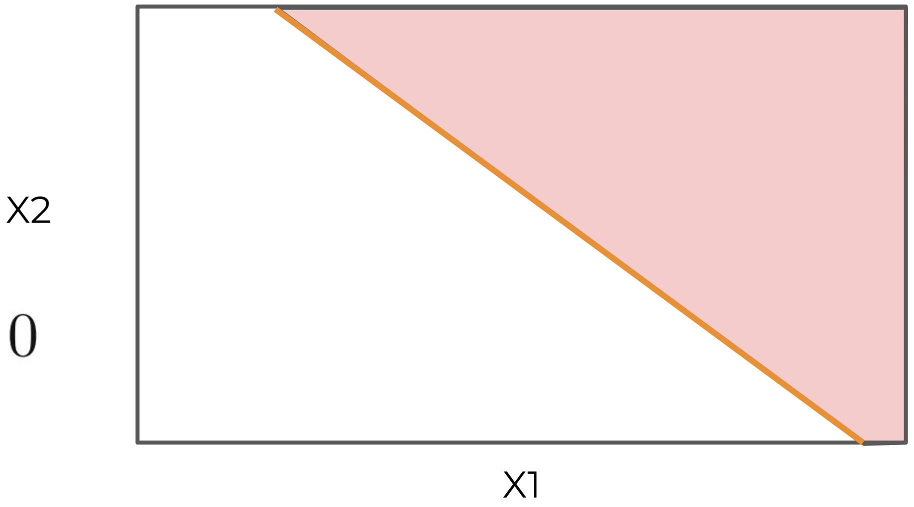
- Separating Hyperplanes



Support Vector Machines

- Separating Hyperplanes

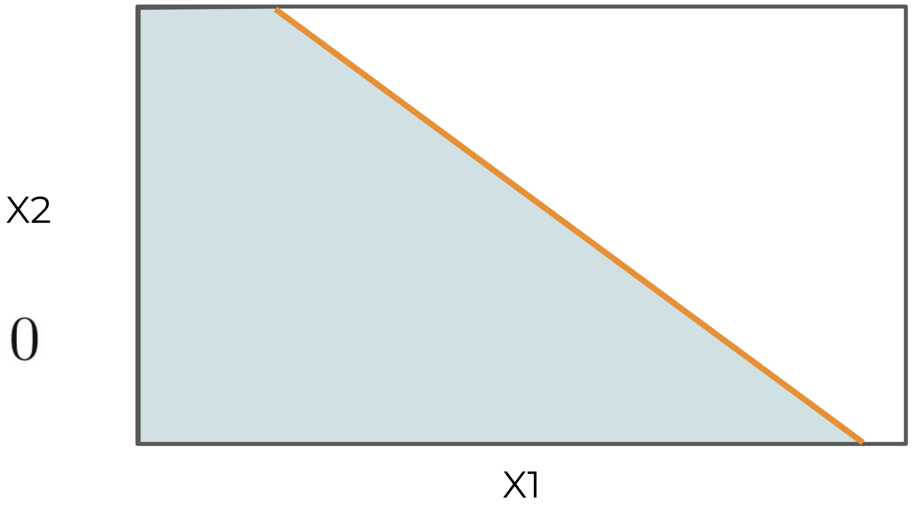
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$



Support Vector Machines

- Separating Hyperplanes

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$

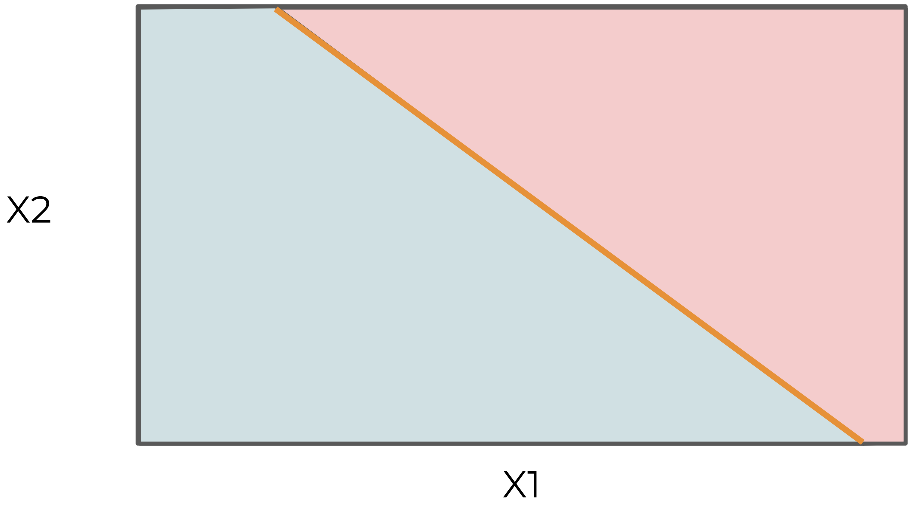


Support Vector Machines

- Separating Hyperplanes

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$

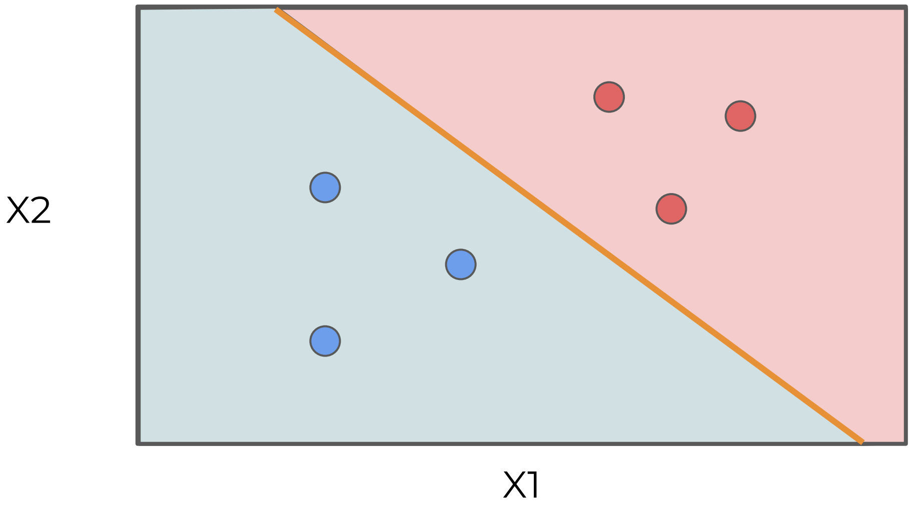
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$



Support Vector Machines

- Data Points

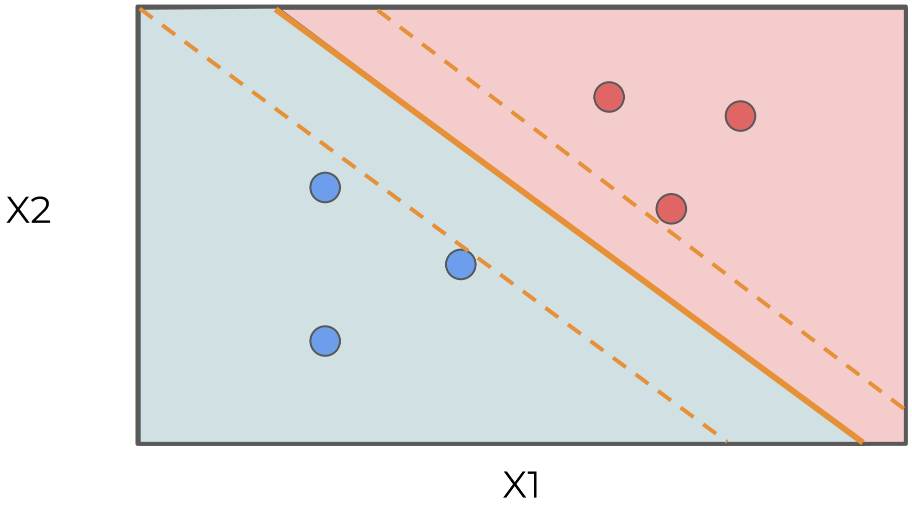
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$



Support Vector Machines

- Max Margin Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

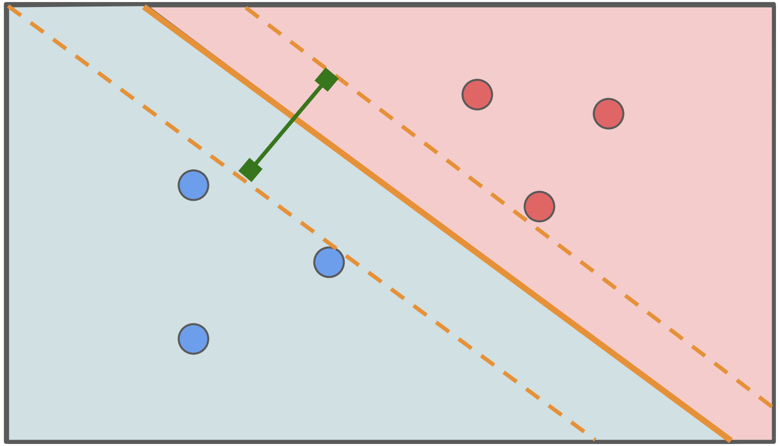


Support Vector Machines

- Max Margin Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; \boxed{M}$$

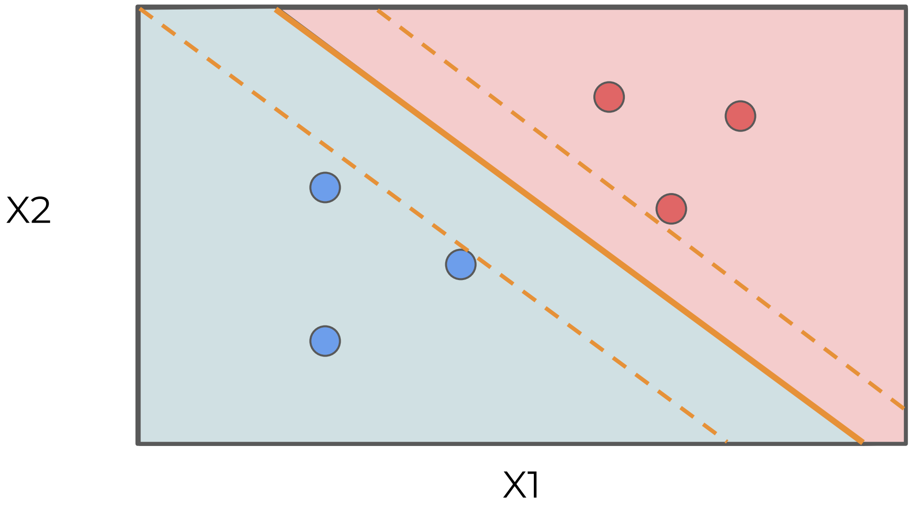
X2



X1

Support Vector Machines

- Max Margin Classifier



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

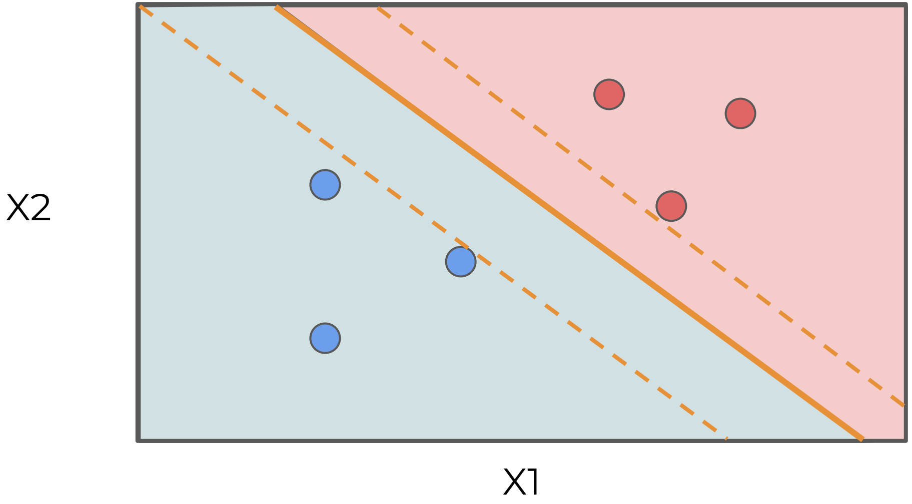
$$\text{subject to} \ \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \ \ \forall\, i = 1, \ldots, n.$$

Support Vector Machines

- Max Margin Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n.$$

Support Vector Machines

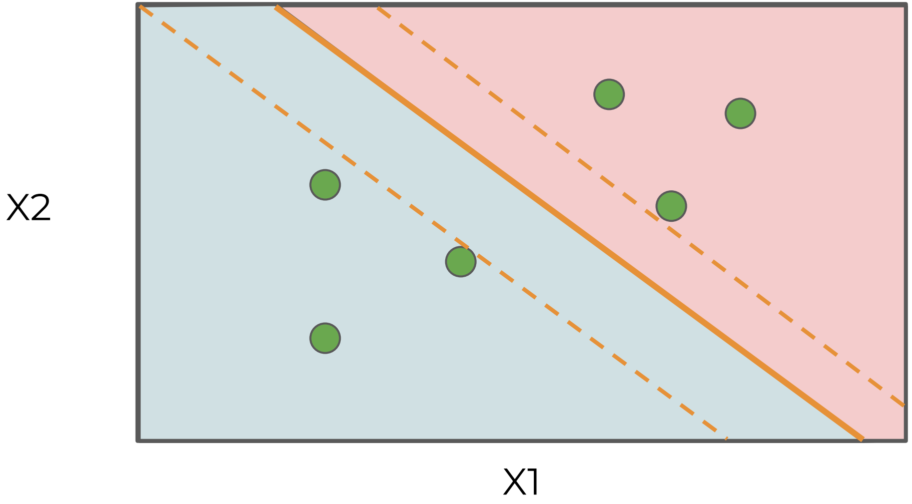
- Max Margin Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \quad \boxed{\forall\, i = 1, \ldots, n.}$$



Support Vector Machines

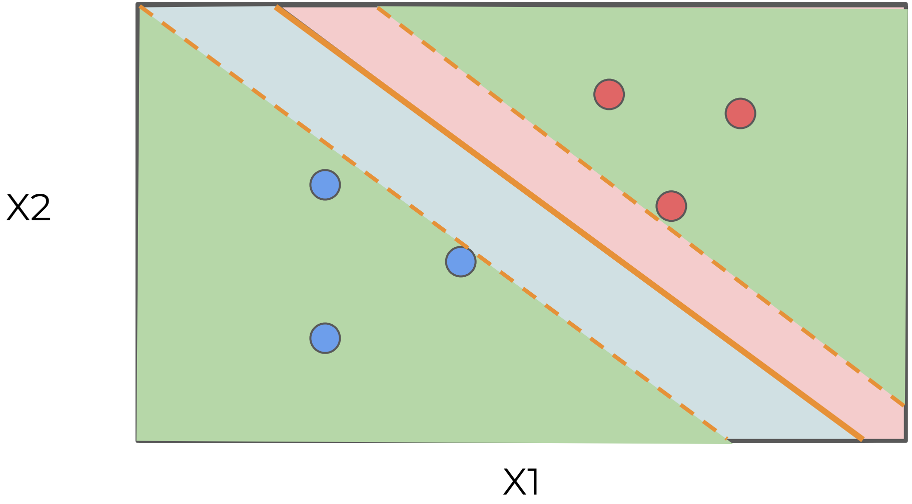
- Max Margin Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

$$\boxed{y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M} \ \forall\, i = 1, \ldots, n.$$



Support Vector Machines

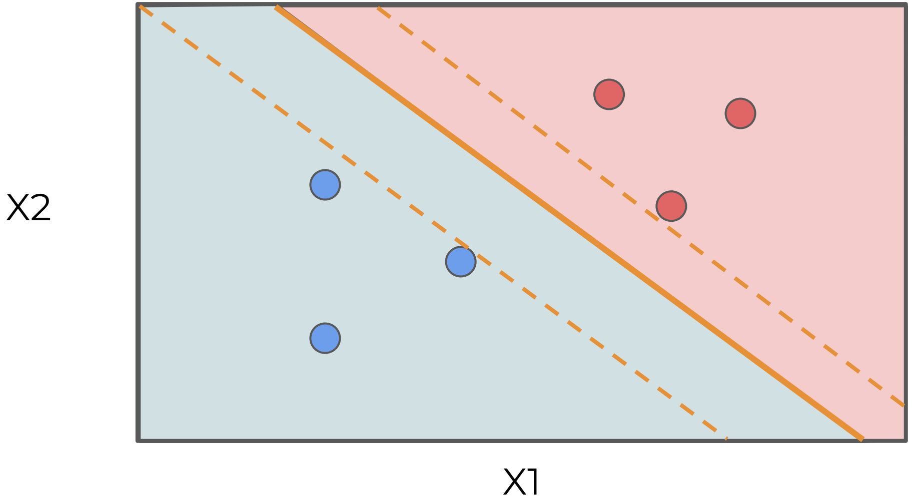
- Max Margin Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

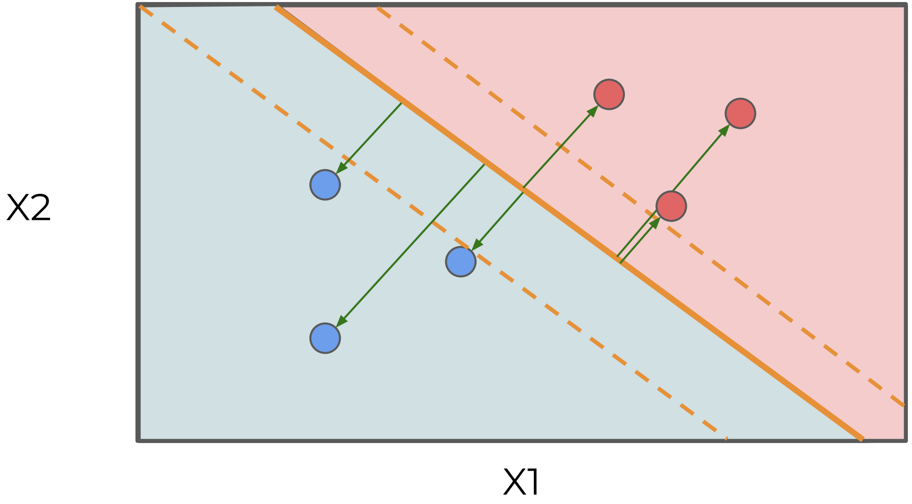
$$\text{subject to} \boxed{\sum_{j=1}^{p} \beta_j^2 = 1}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \ \ \forall\, i = 1, \ldots, n.$$



Support Vector Machines

- Max Margin Classifier



$$\text{subject to} \boxed{\sum_{j=1}^{p} \beta_j^2 = 1}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip})$$

Support Vector Machines

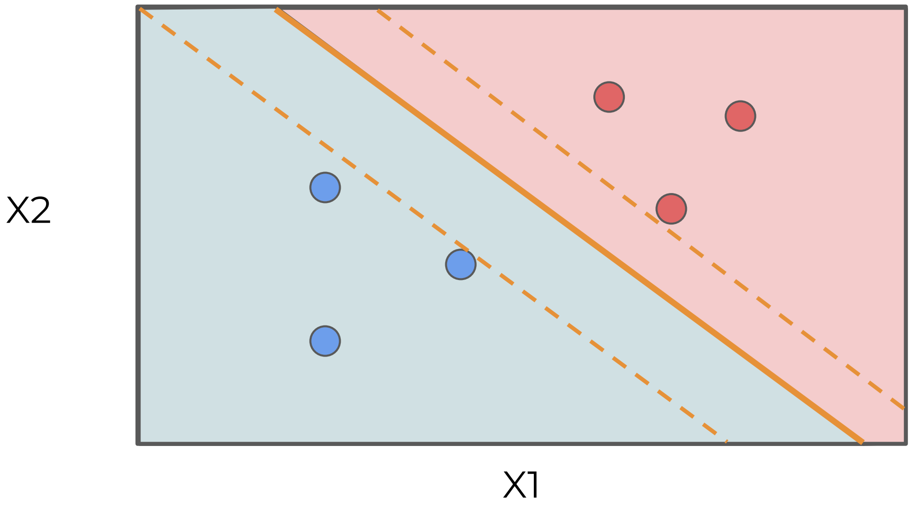
- Max Margin Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

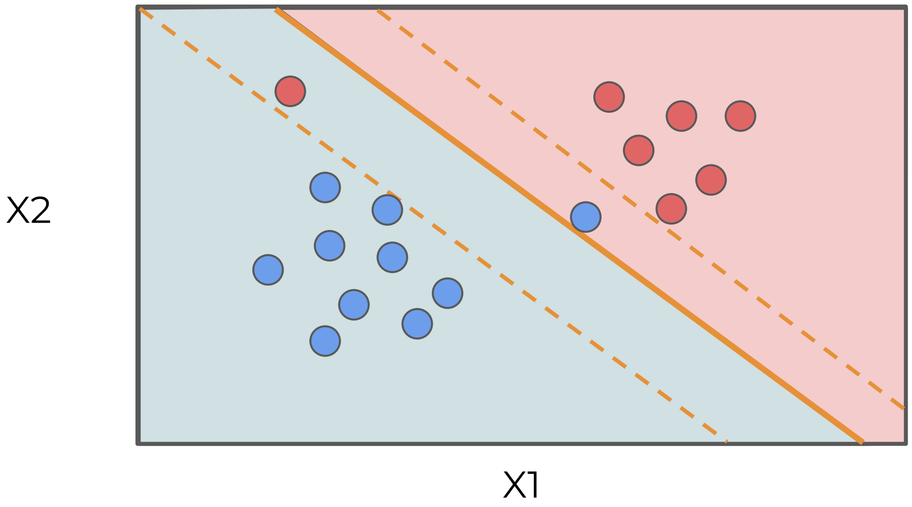
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \ \ \forall\, i = 1, \ldots, n.$$



Support Vector Machines

- Support Vector Classifier

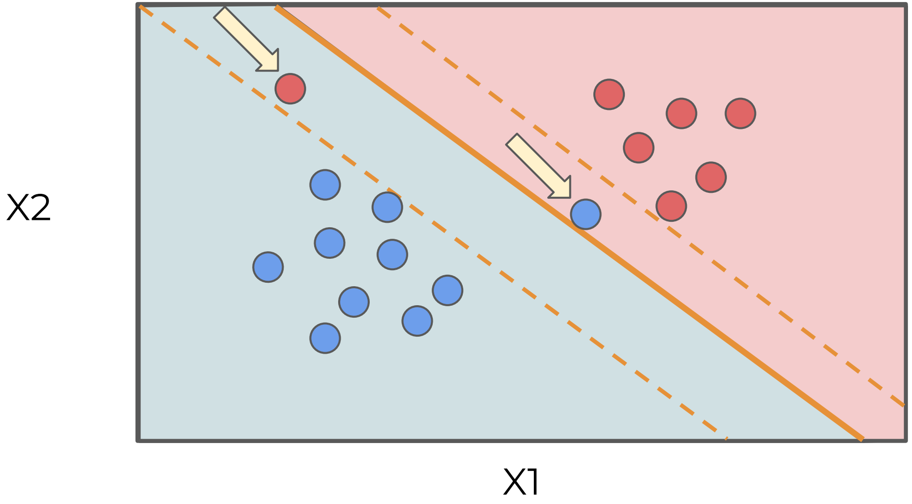
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$



Support Vector Machines

- Support Vector Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

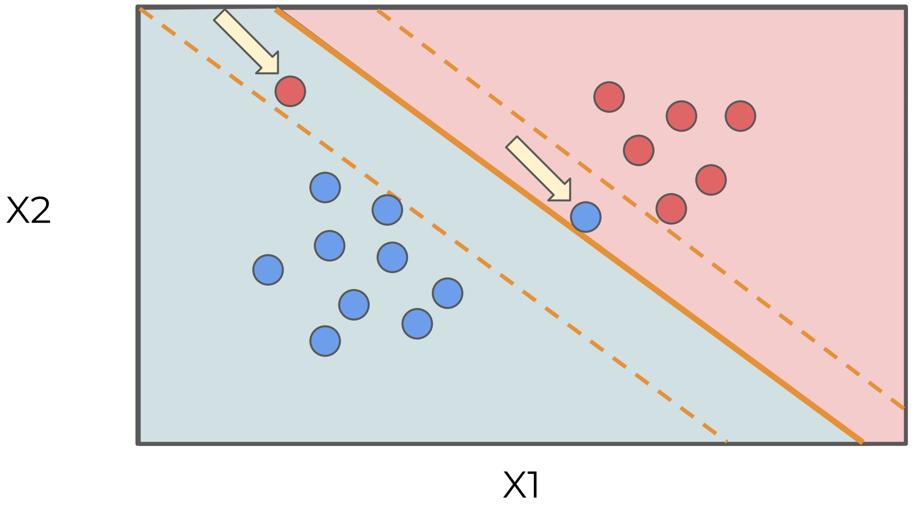


Support Vector Machines

- Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1$$



Support Vector Machines

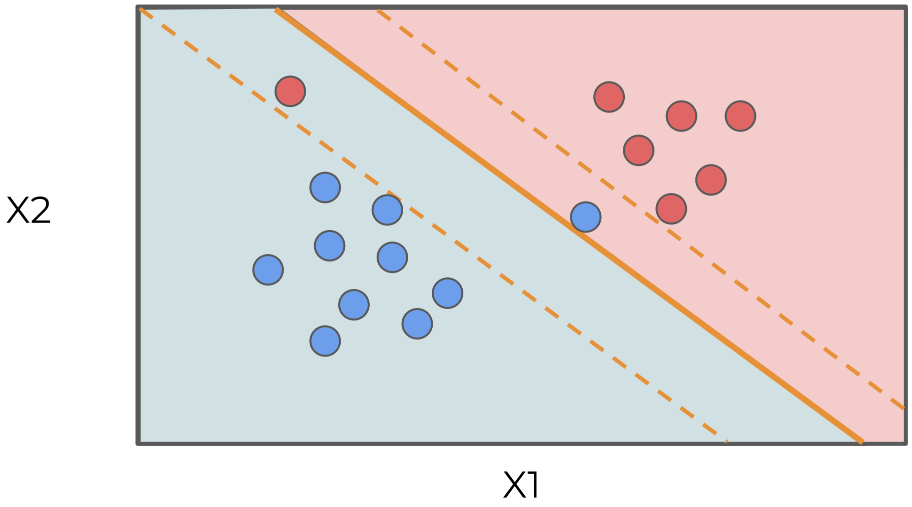
- Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$



Support Vector Machines

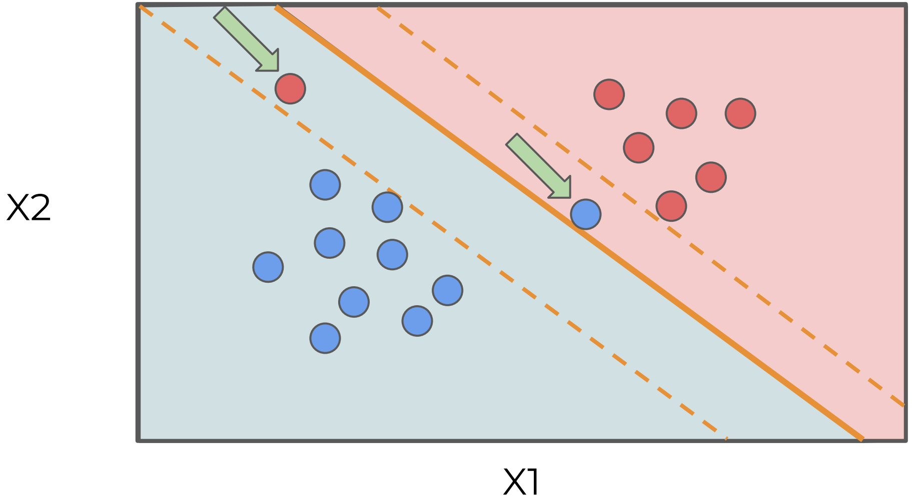
- Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1$$

$$\boxed{\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M\boxed{(1 - \epsilon_i)}$$

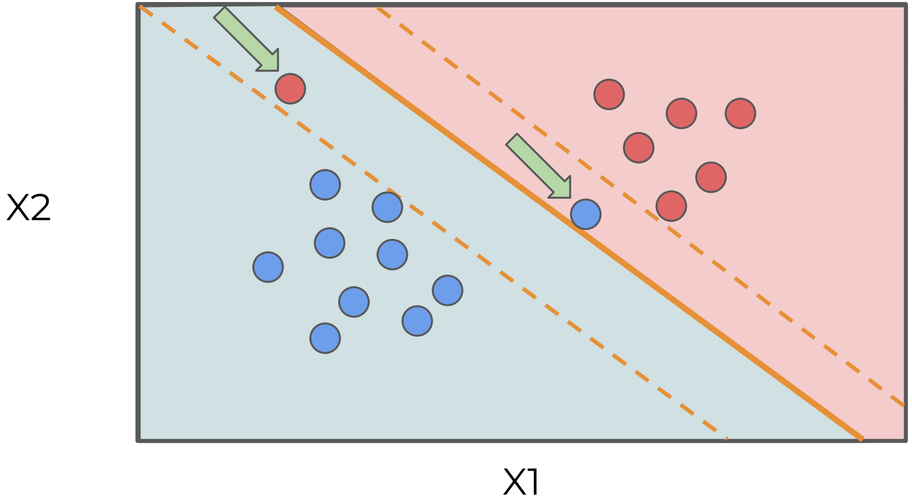


Support Vector Machines

- Note on Scikit-Learn's SVC!

C : *float, default=1.0*

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.



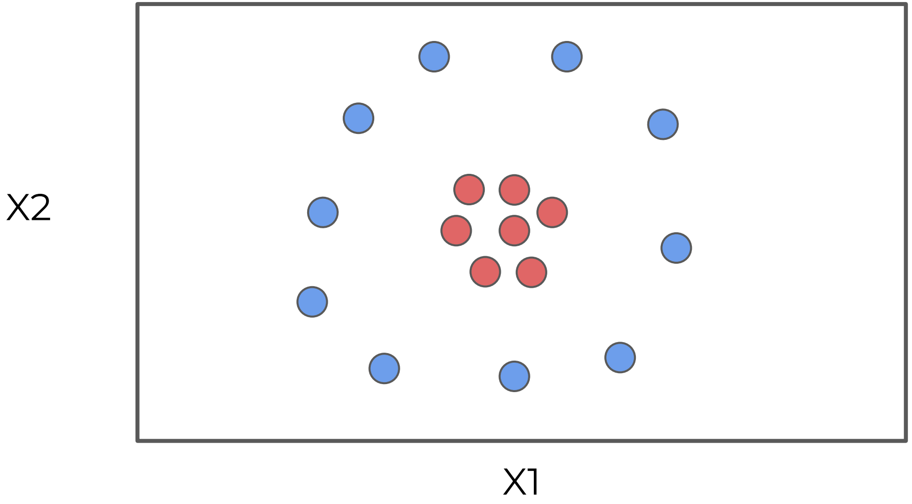
$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

Support Vector Machines

- Support Vector Machines

$$X_1, X_2, \ldots, X_p,$$



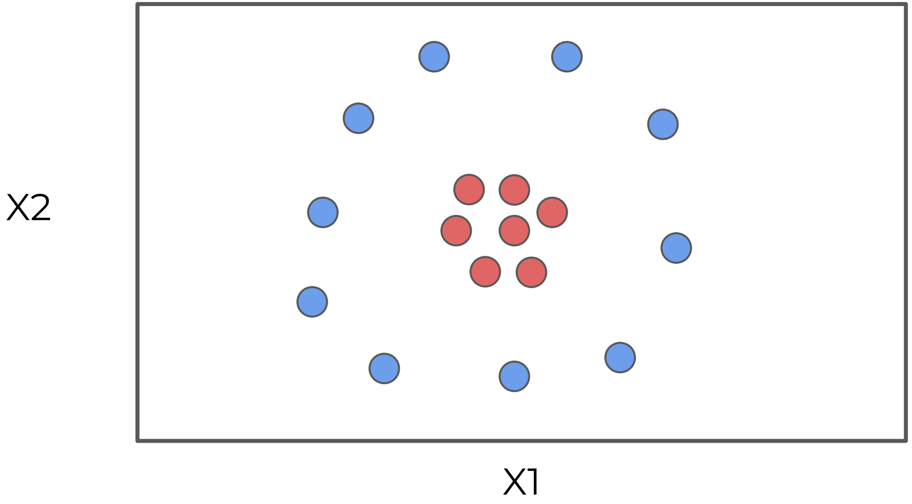
Support Vector Machines

- Support Vector Machines

$$X_1, X_2, \ldots, X_p,$$

$$\Downarrow$$

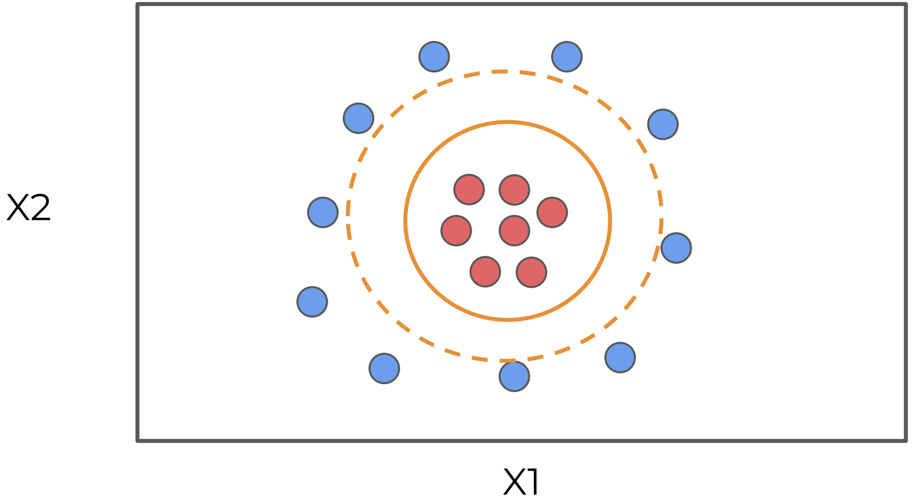
$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$



Support Vector Machines

- Support Vector Machines

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$



Support Vector Machines

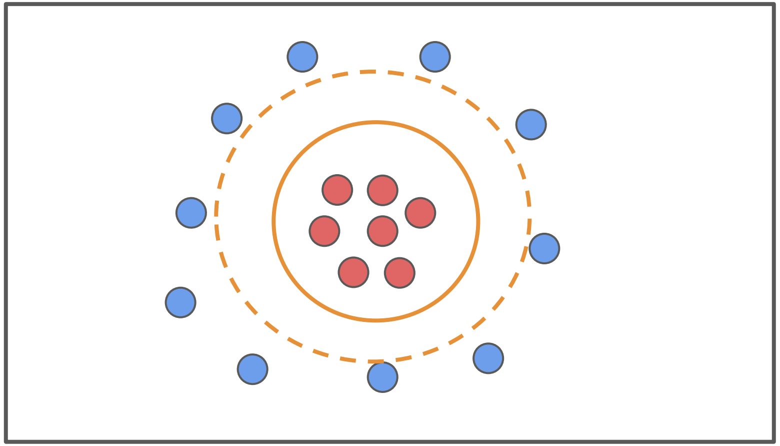
- Support Vector Machines

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \geq M(1-\epsilon_i)$$

$$\sum_{i=1}^{n}\epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1.$$



Support Vector Machines

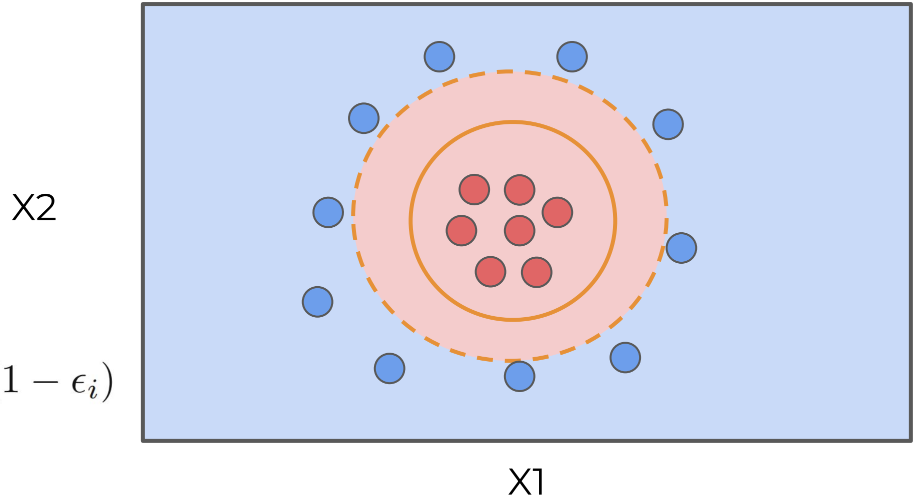
- Support Vector Machines

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \geq M(1-\epsilon_i)$$

$$\sum_{i=1}^{n}\epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1.$$



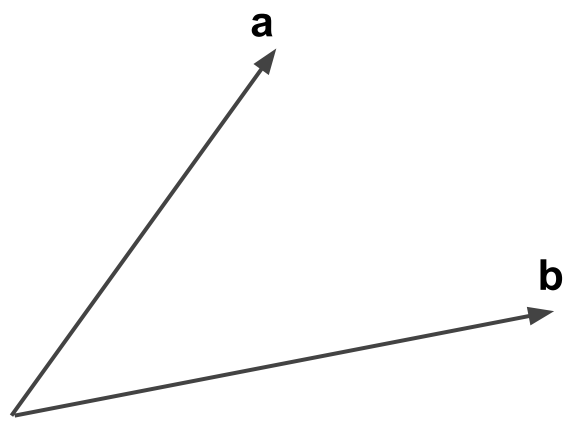
Support Vector Machines

- How to deal with very large feature space?
- As polynomial order grows larger, the number of computations necessary to solve for margins also grows!
- The answer lies in the **kernel trick** which makes use of the **inner product** of vectors, also known as the **dot product**.

Support Vector Machines

- Dot Product

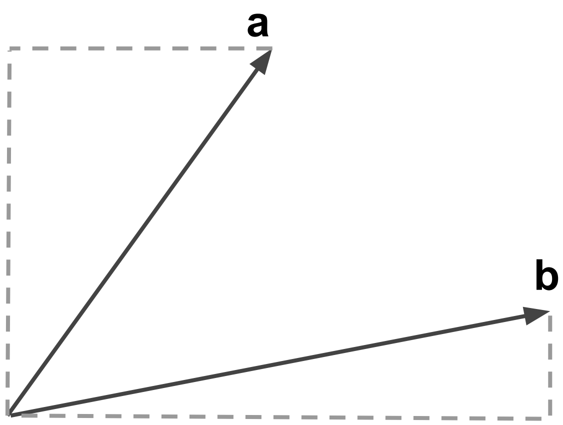
$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$



Support Vector Machines

- Dot Product

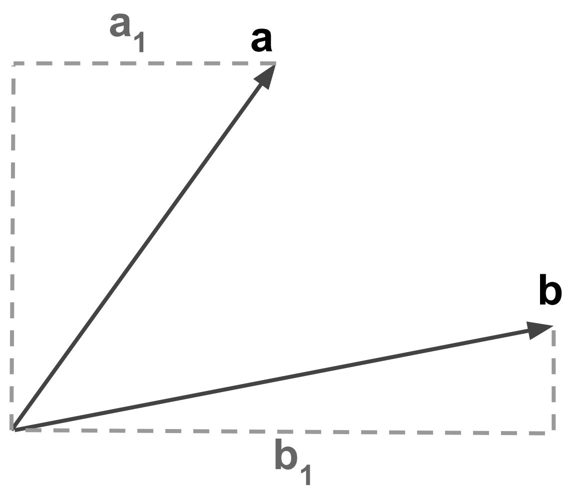
$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$



Support Vector Machines

- Dot Product

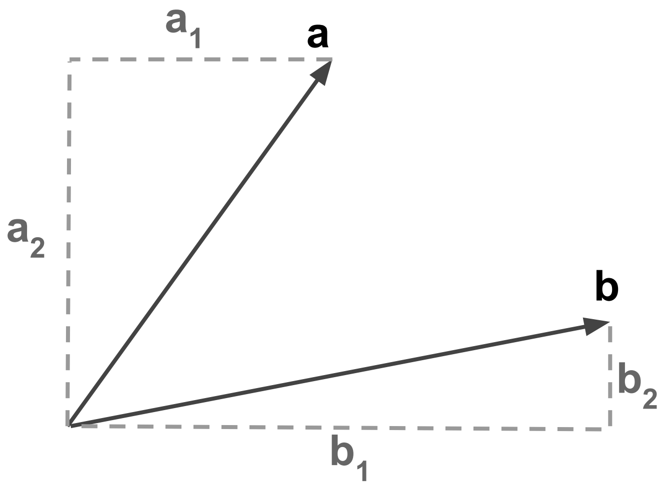
$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$



Support Vector Machines

- Dot Product

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

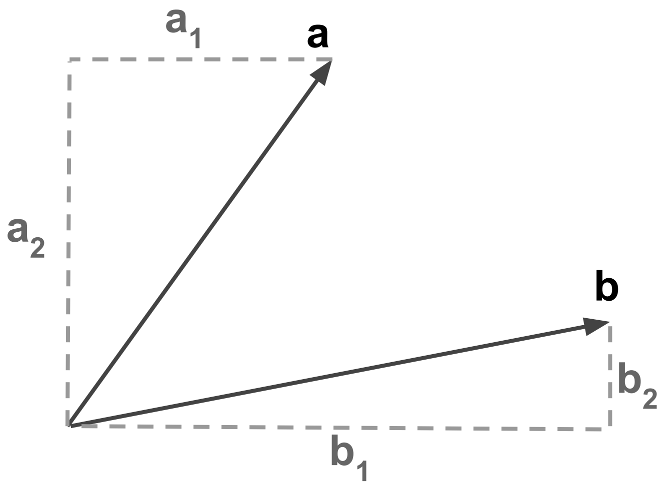


Support Vector Machines

- Dot Product

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

$$a \cdot b = a_1 b_1 + a_2 b_2$$



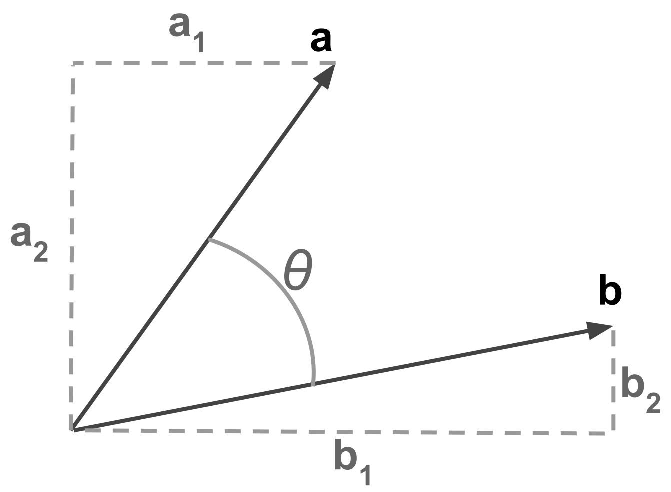
Support Vector Machines

- Dot Product

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

$$a \cdot b = a_1 b_1 + a_2 b_2$$

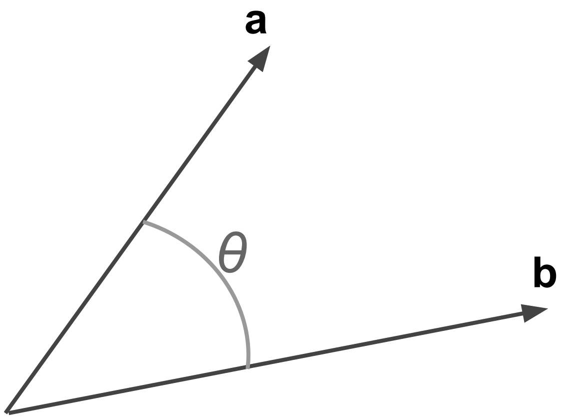
$$a \cdot b = |a||b|cos(\theta)$$



Support Vector Machines

- Notice how the dot product can be thought of as a **similarity** between the vectors.

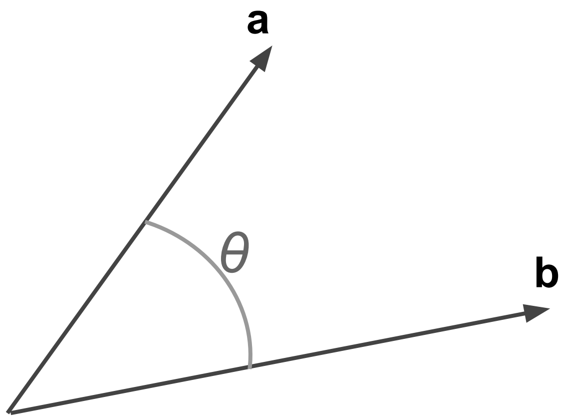
$$a \cdot b = |a||b|cos(\theta)$$



Support Vector Machines

- $\cos(0°) = 1$
- $\cos(90°) = 0$
- $\cos(180°) = -1$

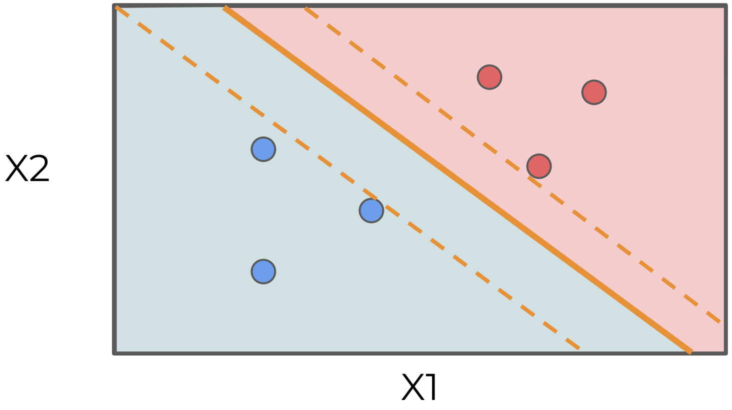
$$a \cdot b = |a||b|cos(\theta)$$



Support Vector Machines

- Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \boxed{\sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle}$$

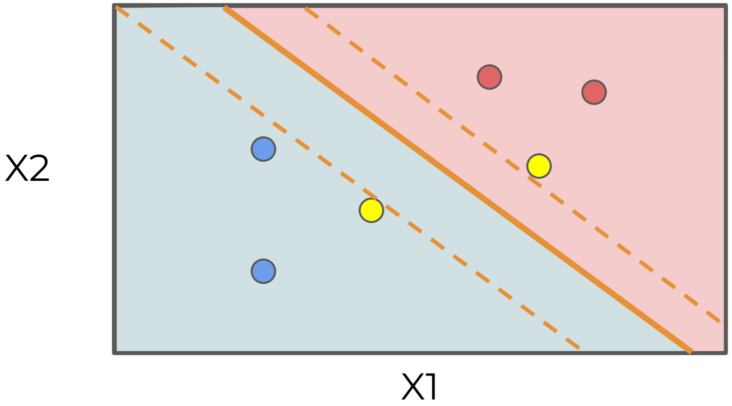


Calculating the inner products of all pairs of training observations

Support Vector Machines

- Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \boxed{\alpha_i} \langle x, x_i \rangle$$



Only non-zero for the support vectors.

Support Vector Machines

- Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

$$\Downarrow$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

Support Vector Machines

- Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{\boxed{i \in \mathcal{S}}} \alpha_i \langle x, x_i \rangle$$

S collection of indices of these support points

Support Vector Machines

- Kernel Function

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

A kernel is a function that quantifies the similarity of two observations.

Support Vector Machines

- Kernel Function

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

Support Vector Machines

- Kernel Function

$$K(x_i, x_{i'}) = \boxed{\sum_{j=1}^{p} x_{ij} x_{i'j}}$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{\langle x, x_i \rangle}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Support Vector Machines

- Kernel Function

$$\boxed{K(x_i, x_{i'})} = \sum_{j=1}^{p} x_{ij} x_{i'j} \qquad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{K(x, x_i)}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Support Vector Machines

- Polynomial Kernel

$$\boxed{K(x_i, x_{i'})} = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \qquad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{\langle x, x_i \rangle}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Support Vector Machines

- Radial Basis Kernel

$$\boxed{K(x_i, x_{i'})} = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{\langle x, x_i \rangle}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Support Vector Machines

- The use of **kernels** as a replacement is known as the **kernel trick**.
- Kernels allow us to avoid computations in the enlarged feature space, by only needing to perform computations for each distinct pair of training points (details in 9.3.2 in ISLR).

Support Vector Machines

- Intuitively we've already seen inner products act as a measurement of similarity between vectors.
- The use of kernels can be thought of as a measure of similarity between the original feature space and the enlarged feature space.

Support Vector Machines

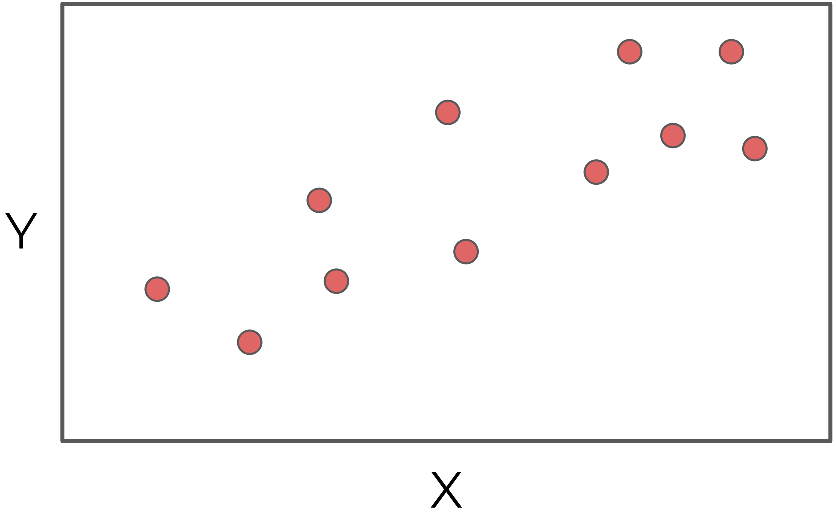
- Now that we understand the theory and intuition behind SVMs, let’s move on to actually using them with code!

Support Vector Machines

Regression with Scikit-Learn

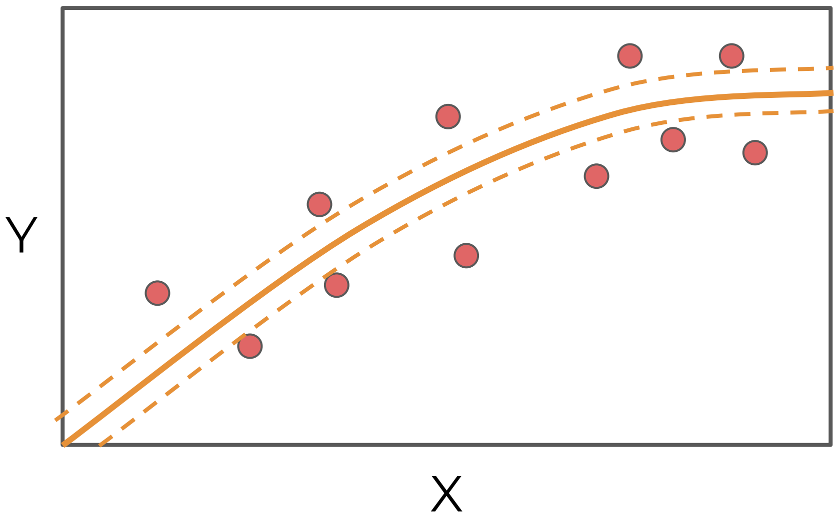
Support Vector Machines

- Support Vector Regression



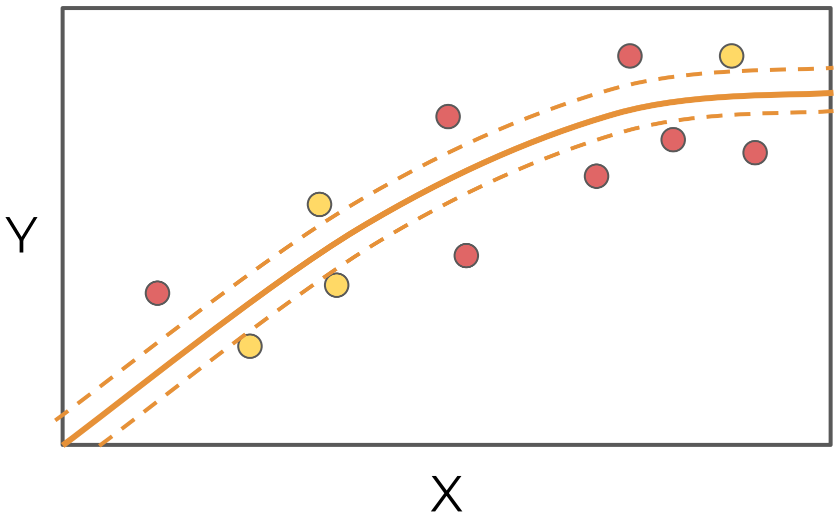
Support Vector Machines

- Support Vector Regression



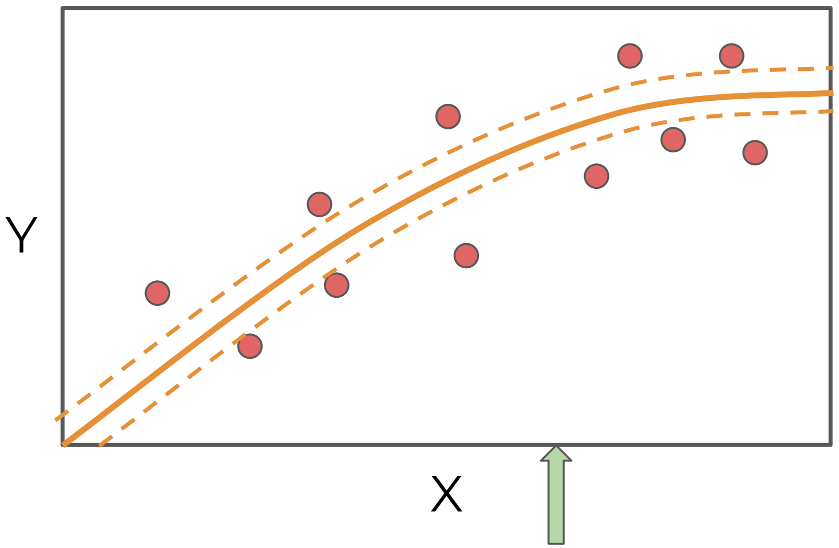
Support Vector Machines

- Support Vector Regression



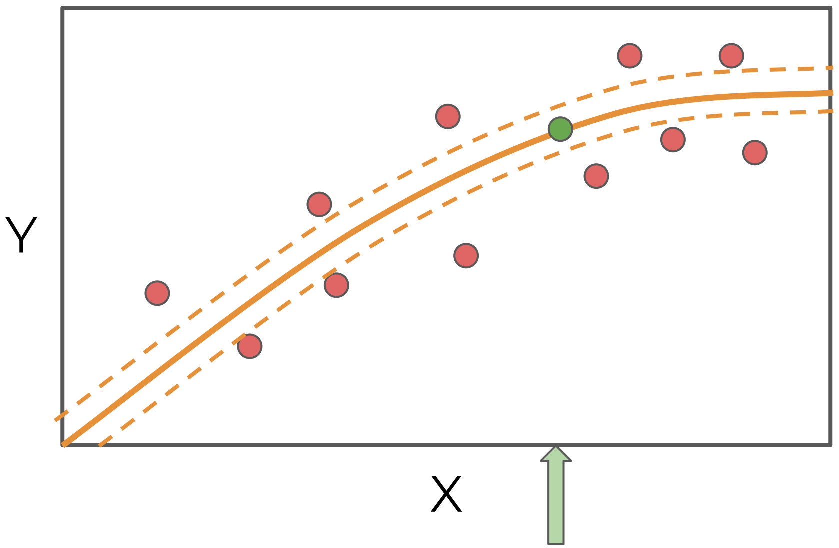
Support Vector Machines

- Support Vector Regression



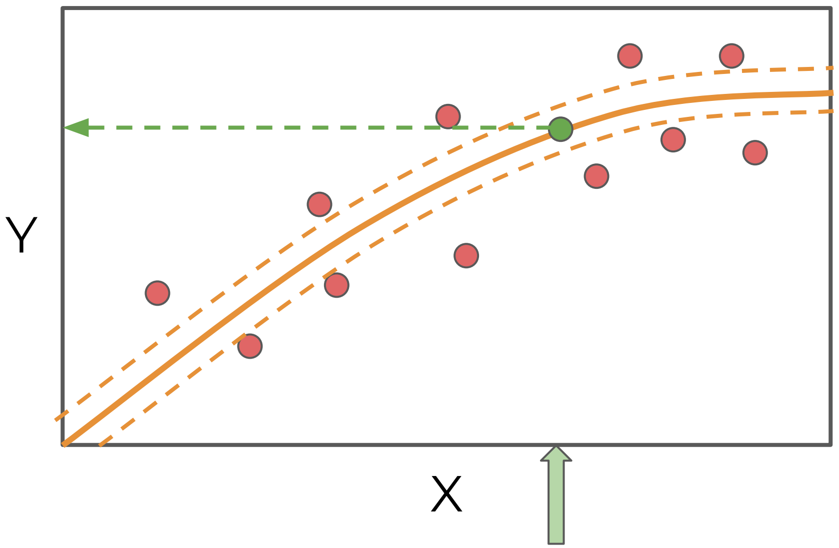
Support Vector Machines

- Support Vector Regression



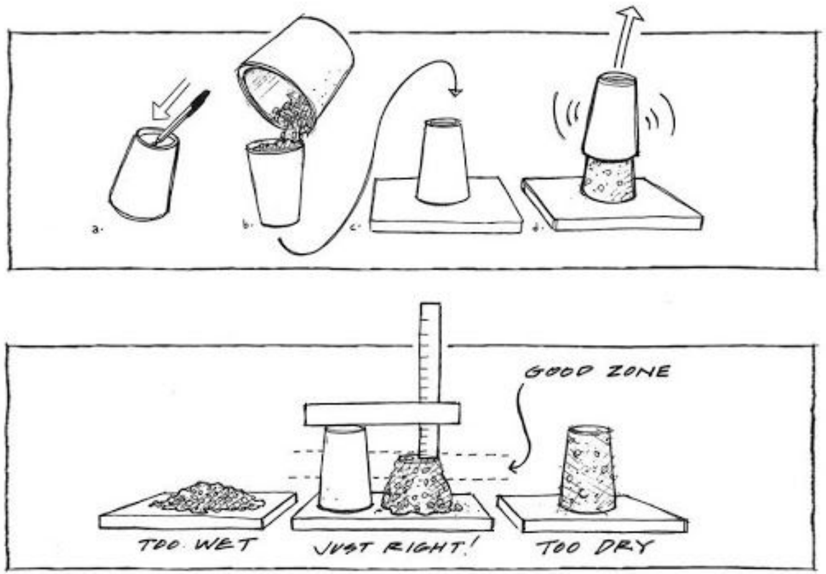
Support Vector Machines

- Support Vector Regression



Support Vector Machines

- Concrete Slump Test

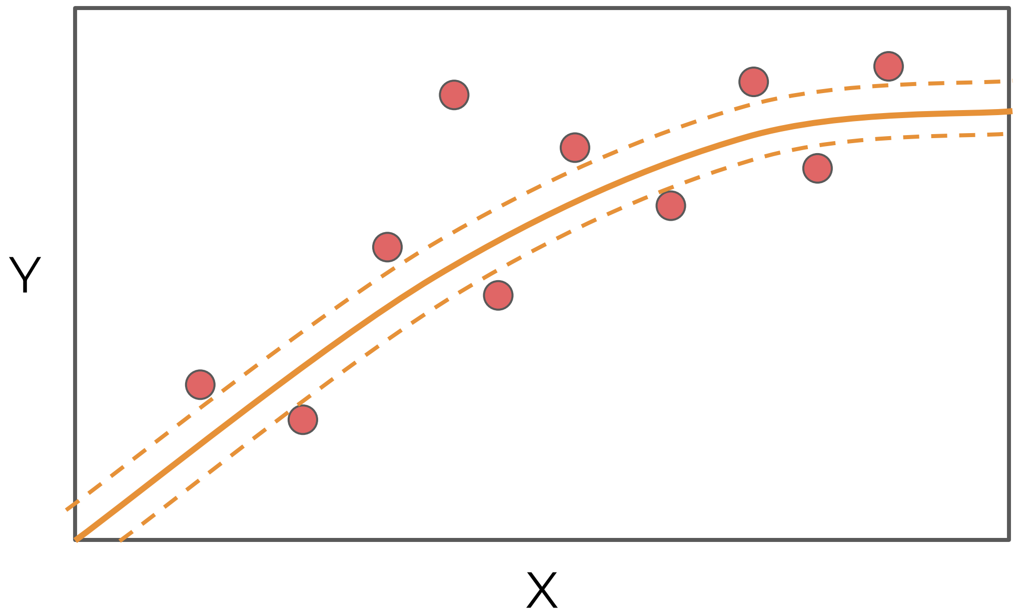


Support Vector Machines

- Concrete Slump Test

Support Vector Machines

- Support Vector Regression



Support Vector Machines

- Support Vector Regression