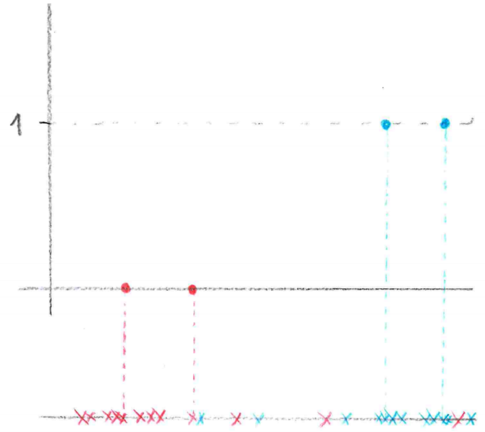
Logistic Regression

What about classification using regression?

Binary classification: Desired outputs 0 and 1

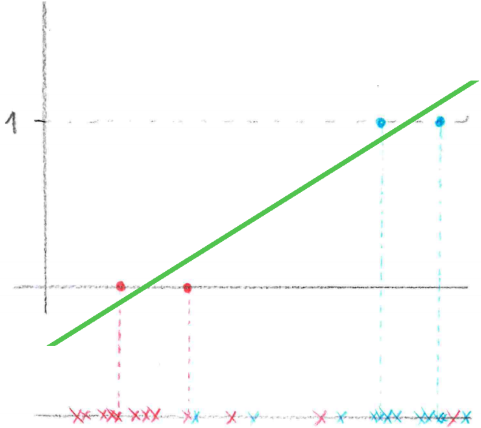
... we want to capture the probability distribution of the classes



What about classification using regression?

Binary classification: Desired outputs 0 and 1

... we want to capture the probability distribution of the classes

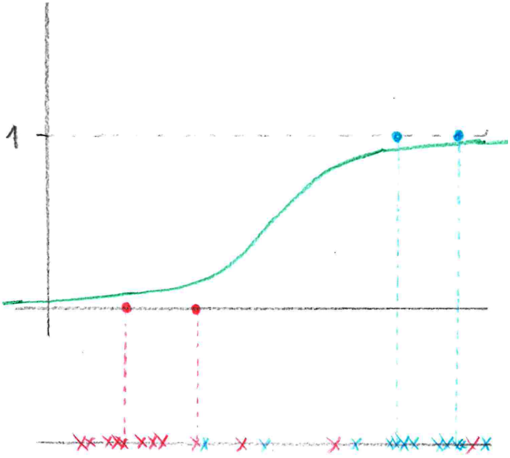


... does not capture the probability well (it is not probability at all)

What about classification using regression?

Binary classification: Desired outputs 0 and 1

... we want to capture the probability distribution of the classes



... logistic sigmoid $\frac{1}{1+e^{-(\vec{w}\cdot\tilde{\mathbf{x}})}}$ is much better!

Logistic Regression

Logistic regression model $h[\vec{w}]$ is determined by a vector of weights $\vec{w} = (w_0, w_1, \ldots, w_n) \in \mathbb{R}^{n+1}$ as follows:

Logistic Regression

Logistic regression model $h[\vec{w}]$ is determined by a vector of weights $\vec{w} = (w_0, w_1, \ldots, w_n) \in \mathbb{R}^{n+1}$ as follows:

Given $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$,

$$h[\vec{w}](\vec{x}) := \frac{1}{1 + e^{-\left(w_0 + \sum_{k=1}^{n} w_k x_k\right)}} = \frac{1}{1 + e^{-\vec{w}\cdot\tilde{\mathbf{x}}}}$$

Here

$$\tilde{\mathbf{x}} = (x_0, x_1, \ldots, x_n) \quad \text{where } x_0 = 1$$

is the *augmented feature vector*.

But what is the meaning of the sigmoid?

The model gives probability $h[\vec{w}](\vec{x})$ of the class 1 given an input $\vec{x}$.
But why do we model such probability using $1/(1+e^{-\vec{w}\cdot\tilde{\mathbf{x}}})$??

But what is the meaning of the sigmoid?

The model gives probability $h[\vec{w}](\vec{x})$ of the class 1 given an input $\vec{x}$. But why do we model such probability using $1/(1 + e^{-\vec{w}\cdot\tilde{\mathbf{x}}})$??

Denote by $\bar{h}$ the probability $P(Y = 1 \mid X = \vec{x})$, i.e., the "true" probability of the class 1 given features $\vec{x}$.

The probability $\bar{h}$ cannot be easily modeled using a linear function (the probabilities are between 0 and 1).

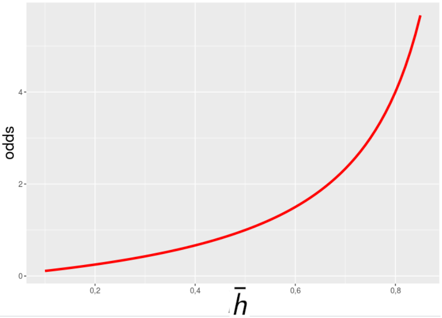
But what is the meaning of the sigmoid?

The model gives probability $h[\vec{w}](\vec{x})$ of the class 1 given an input $\vec{x}$. But why do we model such probability using $1/(1+e^{-\vec{w}\cdot\tilde{\mathbf{x}}})$??

Denote by $\bar{h}$ the probability $P(Y=1 \mid X=\vec{x})$, i.e., the "true" probability of the class 1 given features $\vec{x}$.

What about odds of the class 1?

$$odds(\bar{h}) = \bar{h}/(1-\bar{h})$$



Better, at least it is unbounded on one side ...

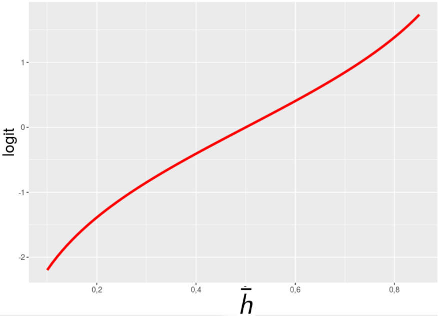
But what is the meaning of the sigmoid?

The model gives probability $h[\vec{w}](\vec{x})$ of the class 1 given an input $\vec{x}$. But why do we model such probability using $1/(1 + e^{-\vec{w}\cdot\tilde{\mathbf{x}}})$??

Denote by $\bar{h}$ the probability $P(Y = 1 \mid X = \vec{x})$, i.e., the "true" probability of the class 1 given features $\vec{x}$.

What about log odds (aka logit) of the class 1?

$$logit(\bar{h}) = \log(\bar{h}/(1 - \bar{h}))$$



Looks almost linear, at least for probabilities not too close to 0 or 1 ...

But what is the meaning of the sigmoid?

Assume that $\bar{h}$ is the actual probability of the class 1 for an "object" with features $\vec{x} \in \mathbb{R}^n$. Put

$$\log(\bar{h}/(1-\bar{h})) = \vec{w} \cdot \tilde{\mathbf{x}}$$

But what is the meaning of the sigmoid?

Assume that $\bar{h}$ is the actual probability of the class 1 for an "object" with features $\vec{x} \in \mathbb{R}^n$. Put

$$\log(\bar{h}/(1-\bar{h})) = \vec{w} \cdot \tilde{\mathbf{x}}$$

Then

$$\log((1-\bar{h})/\bar{h})) = -\vec{w} \cdot \tilde{\mathbf{x}}$$

But what is the meaning of the sigmoid?

Assume that $\bar{h}$ is the actual probability of the class 1 for an "object" with features $\vec{x} \in \mathbb{R}^n$. Put

$$\log(\bar{h}/(1-\bar{h})) = \vec{w} \cdot \tilde{\mathbf{x}}$$

Then

$$\log((1-\bar{h})/\bar{h})) = -\vec{w} \cdot \tilde{\mathbf{x}}$$

and

$$(1-\bar{h})/\bar{h} = e^{-\vec{w} \cdot \tilde{\mathbf{x}}}$$

But what is the meaning of the sigmoid?

Assume that $\bar{h}$ is the actual probability of the class 1 for an "object" with features $\vec{x} \in \mathbb{R}^n$. Put

$$\log(\bar{h}/(1-\bar{h})) = \vec{w} \cdot \tilde{\mathbf{x}}$$

Then

$$\log((1-\bar{h})/\bar{h})) = -\vec{w} \cdot \tilde{\mathbf{x}}$$

and

$$(1-\bar{h})/\bar{h} = e^{-\vec{w} \cdot \tilde{\mathbf{x}}}$$

and

$$\bar{h} = \frac{1}{1+e^{-\vec{w} \cdot \tilde{\mathbf{x}}}} = h[\vec{w}](\vec{x})$$

If we model log odds using a linear function, the probability is obtained by applying the logistic sigmoid on the result of the linear function.

Logistic Regression

- Given a set D of training samples:

$$D = \{(\vec{x}_1, c_1), (\vec{x}_2, c_2), \ldots, (\vec{x}_p, c_p)\}$$

Here $\vec{x}_k = (x_{k1} \ldots, x_{kn}) \in \mathbb{R}^n$ and $c_k \in \{0, 1\}$.

Logistic Regression

- Given a set D of training samples:

$$D = \{(\vec{x}_1, c_1), (\vec{x}_2, c_2), \ldots, (\vec{x}_p, c_p)\}$$

Here $\vec{x}_k = (x_{k1} \ldots, x_{kn}) \in \mathbb{R}^n$ and $c_k \in \{0, 1\}$.

Recall that $h[\vec{w}](\vec{x}_k) = 1 \,/\, \left(1 + e^{-\vec{w} \cdot \tilde{\mathbf{x}}_k}\right)$ where $\tilde{\mathbf{x}}_k = (x_{k0}, x_{k1} \ldots, x_{kn})$, here $x_{k0} = 1$

Our goal: Find $\vec{w}$ such that for every $k = 1, \ldots, p$ we have that $h[\vec{w}](\vec{x}_k) \approx c_k$

Logistic Regression

- Given a set D of training samples:

$$D = \{(\vec{x}_1, c_1), (\vec{x}_2, c_2), \dots, (\vec{x}_p, c_p)\}$$

Here $\vec{x}_k = (x_{k1} \dots, x_{kn}) \in \mathbb{R}^n$ and $c_k \in \{0, 1\}$.

Recall that $h[\vec{w}](\vec{x}_k) = 1 \,/\, \left(1 + e^{-\vec{w}\cdot\tilde{\mathbf{x}}_k}\right)$ where
$\tilde{\mathbf{x}}_k = (x_{k0}, x_{k1} \dots, x_{kn})$, here $x_{k0} = 1$

Our goal: Find $\vec{w}$ such that for every $k = 1, \dots, p$ we have that $h[\vec{w}](\vec{x}_k) \approx c_k$

- **Binary Cross-entropy:**

$$E(\vec{w}) = -\sum_{k=1}^{p} c_k \log(h[\vec{w}](\vec{x}_k)) + (1 - c_k)\log(1 - h[\vec{w}](\vec{x}_k))$$

Gradient of the Error Function

Consider the **gradient** of the error function:

$$\nabla E(\vec{w}) = \left(\frac{\partial E}{\partial w_0}(\vec{w}), \ldots, \frac{\partial E}{\partial w_n}(\vec{w})\right) = \sum_{k=1}^{p} \left(h[\vec{w}](\vec{x}_k) - c_k\right) \cdot \tilde{\mathbf{x}}_k$$

Fact 1

If $\nabla E(\vec{w}) = \vec{0} = (0, \ldots, 0)$*, then* $\vec{w}$ *is a global minimum of* E*.*

This follows from the fact that E is convex.

Using the squared error with the logistic sigmoid would lead to a non-convex error with several minima!

Logistic Regression – Learning

Gradient Descent:

- Weights $\vec{w}^{(0)}$ are initialized randomly close to $\vec{0}$.

Logistic Regression – Learning

Gradient Descent:

- Weights $\vec{w}^{(0)}$ are initialized randomly close to $\vec{0}$.
- In $(t+1)$-th step, $\vec{w}^{(t+1)}$ is computed as follows:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \varepsilon \cdot \nabla E(\vec{w}^{(t)})$$

Logistic Regression – Learning

Gradient Descent:

- Weights $\vec{w}^{(0)}$ are initialized randomly close to $\vec{0}$.
- In $(t+1)$-th step, $\vec{w}^{(t+1)}$ is computed as follows:

$$\begin{aligned} \vec{w}^{(t+1)} &= \vec{w}^{(t)} - \varepsilon \cdot \nabla E(\vec{w}^{(t)}) \\ &= \vec{w}^{(t)} - \varepsilon \cdot \sum_{k=1}^{p} \left(h[\vec{w}^{(t)}](\vec{x}_k) - c_k \right) \cdot \tilde{\mathbf{x}}_k \end{aligned}$$

Here $0 < \varepsilon \leq 1$ is the learning rate.

Note that the algorithm is almost similar to the batch perceptron algorithm!

Logistic Regression – Learning

Gradient Descent:

- Weights $\vec{w}^{(0)}$ are initialized randomly close to $\vec{0}$.
- In $(t+1)$-th step, $\vec{w}^{(t+1)}$ is computed as follows:

$$\begin{aligned}\vec{w}^{(t+1)} &= \vec{w}^{(t)} - \varepsilon \cdot \nabla E(\vec{w}^{(t)}) \\ &= \vec{w}^{(t)} - \varepsilon \cdot \sum_{k=1}^{p} \left(h[\vec{w}^{(t)}](\vec{x}_k) - c_k \right) \cdot \tilde{\mathbf{x}}_k\end{aligned}$$

  Here $0 < \varepsilon \leq 1$ is the learning rate.

Note that the algorithm is almost similar to the batch perceptron algorithm!

Proposition

For sufficiently small $\varepsilon > 0$, the sequence $\vec{w}^{(0)}, \vec{w}^{(1)}, \vec{w}^{(2)}, \ldots$ converges (in a component-wise manner) to the global minimum of the error function E.

Logistic Regression - Using the Trained Model

We have already trained our logistic regression model, i.e., we have a vector of weights $\vec{w} = (w_0, w_1, \ldots, w_n)$.

Logistic Regression - Using the Trained Model

We have already trained our logistic regression model, i.e., we have a vector of weights $\vec{w} = (w_0, w_1, \ldots, w_n)$.

The model is the function $h[\vec{w}]$ which for a given feature vector $\vec{x} = (x_1, \ldots, x_n)$ returns the probability

$$h[\vec{w}](\vec{x}) = \frac{1}{1 + e^{-\left(w_0 + \sum_{k=1}^{n} w_k x_k\right)}}$$

that $\vec{x}$ belongs to the class 1.

Logistic Regression - Using the Trained Model

We have already trained our logistic regression model, i.e., we have a vector of weights $\vec{w} = (w_0, w_1, \ldots, w_n)$.

The model is the function $h[\vec{w}]$ which for a given feature vector $\vec{x} = (x_1, \ldots, x_n)$ returns the probability

$$h[\vec{w}](\vec{x}) = \frac{1}{1 + e^{-\left(w_0 + \sum_{k=1}^{n} w_k x_k\right)}}$$

that $\vec{x}$ belongs to the class 1.

To decide whether a given $\vec{x}$ belongs to the class 1 we use $h[\vec{w}]$ as a Bayes classifier: Assign $\vec{x}$ to the class 1 iff $h[\vec{w}](\vec{x}) \geq 1/2$.

Other thresholds can also be used depending on the application and properties of the model. In such a case, given a threshold $\xi \in [0, 1]$, assign $\vec{x}$ to the class 1 iff $h[\vec{w}](\vec{x}) \geq \xi$.

Maximum Likelihood vs Cross-entropy (Dim 1)

Fix a training set $D = \{(x_1, c_1), (x_2, c_2), \ldots, (x_p, c_p)\}$

Maximum Likelihood vs Cross-entropy (Dim 1)

Fix a training set $D = \{(x_1, c_1), (x_2, c_2), \ldots, (x_p, c_p)\}$
Generate a sequence $c'_1, \ldots, c'_p \in \{0, 1\}^p$ where each c'_k has been generated independently by the Bernoulli trial generating 1 with probability

$$h[w_0, w_1](x_k) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_k)}}$$

and 0 otherwise.

Here w_0, w_1 are **unknown weights**.

Maximum Likelihood vs Cross-entropy (Dim 1)

Fix a training set $D = \{(x_1, c_1), (x_2, c_2), \ldots, (x_p, c_p)\}$
Generate a sequence $c'_1, \ldots, c'_p \in \{0, 1\}^p$ where each c'_k has been generated independently by the Bernoulli trial generating 1 with probability

$$h[w_0, w_1](x_k) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_k)}}$$

and 0 otherwise.

Here w_0, w_1 are **unknown weights**.

How "probable" is it to generate the correct classes $c_1, \ldots, c_p$?

Maximum Likelihood vs Cross-entropy (Dim 1)

Fix a training set $D = \{(x_1, c_1), (x_2, c_2), \ldots, (x_p, c_p)\}$
Generate a sequence $c'_1, \ldots, c'_p \in \{0,1\}^p$ where each c'_k has been generated independently by the Bernoulli trial generating 1 with probability

$$h[w_0, w_1](x_k) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_k)}}$$

and 0 otherwise.

Here w_0, w_1 are **unknown weights**.

How "probable" is it to generate the correct classes $c_1, \ldots, c_p$?

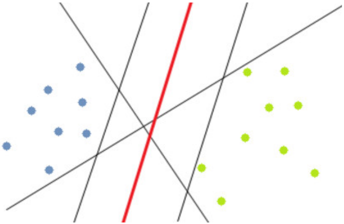
The following conditions are equivalent:

- w_0, w_1 minimize the binary cross-entropy E
- w_0, w_1 maximize the likelihood (i.e., the "probability") of generating the correct values $c_1, \ldots, c_p$ using the above described Bernoulli trials (i.e., that $c'_k = c_k$ for all $k = 1, \ldots, p$)

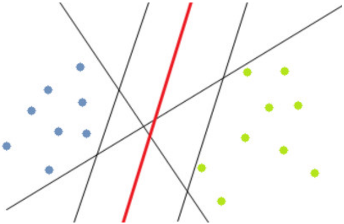
Note that the above equivalence is a property of the cross-entropy and is not dependent on the "implementation" of $h[w_0, w_1](x_k)$ using the logistic sigmoid.

Support Vector Machines (SVM)

SVM Idea – Which Linear Classifier is the Best?



SVM Idea – Which Linear Classifier is the Best?



Benefits of maximum margin:

- Intuitively, the maximum margin is good w.r.t. generalization.
- Only the *support vectors* (those on the margin) matter; others can, in principle, be ignored.

Support Vector Machines (SVM)

Notation:

- $\vec{w} = (w_0, w_1, \ldots, w_n)$ a vector of weights,

Support Vector Machines (SVM)

Notation:

- $\vec{w} = (w_0, w_1, \ldots, w_n)$ a vector of weights,
- $\underline{\vec{w}} = (w_1, \ldots, w_n)$ a vector of all weights except w_0,

Support Vector Machines (SVM)

Notation:

- $\vec{w} = (w_0, w_1, \ldots, w_n)$ a vector of weights,
- $\underline{\vec{w}} = (w_1, \ldots, w_n)$ a vector of all weights except w_0,
- $\vec{x} = (x_1, \ldots, x_n)$ a (generic) feature vector.

Support Vector Machines (SVM)

Notation:

- $\vec{w} = (w_0, w_1, \ldots, w_n)$ a vector of weights,
- $\underline{\vec{w}} = (w_1, \ldots, w_n)$ a vector of all weights except w_0,
- $\vec{x} = (x_1, \ldots, x_n)$ a (generic) feature vector.
- $\tilde{\mathbf{x}} = (x_0, x_1, \ldots, x_n)$ an augmented feature vector where $x_0 = 1$.

Support Vector Machines (SVM)

Notation:

- $\vec{w} = (w_0, w_1, \ldots, w_n)$ a vector of weights,
- $\underline{\vec{w}} = (w_1, \ldots, w_n)$ a vector of all weights except w_0,
- $\vec{x} = (x_1, \ldots, x_n)$ a (generic) feature vector.
- $\tilde{\mathbf{x}} = (x_0, x_1, \ldots, x_n)$ an augmented feature vector where $x_0 = 1$.

Consider a linear classifier:

$$h[\vec{w}](\vec{x}) := \begin{cases} 1 & w_0 + \sum_{i=1}^{n} w_i \cdot x_i = \vec{w} \cdot \tilde{\mathbf{x}} \geq 0 \\ -1 & w_0 + \sum_{i=1}^{n} w_i \cdot x_i = \vec{w} \cdot \tilde{\mathbf{x}} < 0 \end{cases}$$

Support Vector Machines (SVM)

Notation:

- $\vec{w} = (w_0, w_1, \ldots, w_n)$ a vector of weights,
- $\underline{\vec{w}} = (w_1, \ldots, w_n)$ a vector of all weights except w_0,
- $\vec{x} = (x_1, \ldots, x_n)$ a (generic) feature vector.
- $\tilde{\mathbf{x}} = (x_0, x_1, \ldots, x_n)$ an augmented feature vector where $x_0 = 1$.

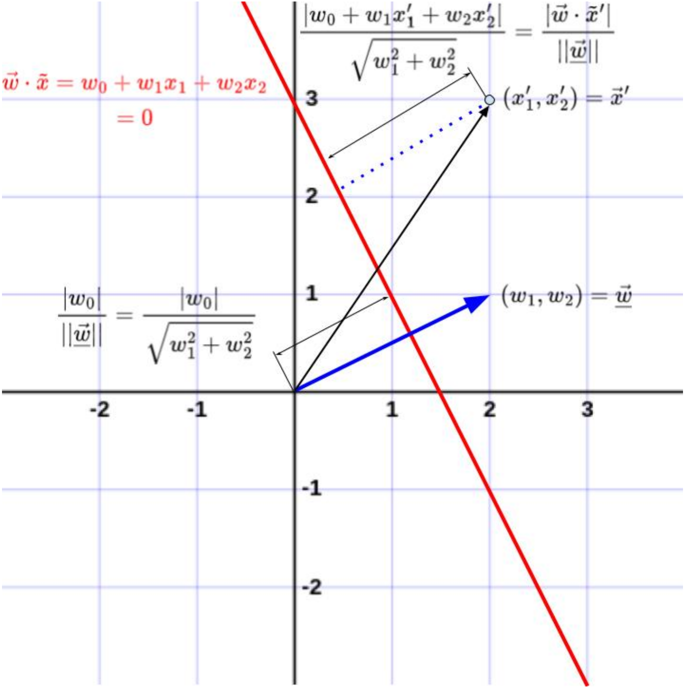
Consider a linear classifier:

$$h[\vec{w}](\vec{x}) := \begin{cases} 1 & w_0 + \sum_{i=1}^{n} w_i \cdot x_i = \vec{w} \cdot \tilde{\mathbf{x}} \geq 0 \\ -1 & w_0 + \sum_{i=1}^{n} w_i \cdot x_i = \vec{w} \cdot \tilde{\mathbf{x}} < 0 \end{cases}$$

The *distance* of $\vec{x}$ from the separating hyperplane determined by $\vec{w}$ is

$$d[\vec{w}](\vec{x}) = \frac{|\vec{w} \cdot \tilde{\mathbf{x}}|}{\|\underline{\vec{w}}\|}$$

Recall that $\vec{w} \cdot \tilde{\mathbf{x}}$ is positive for $\vec{x}$ on the side to which $\underline{\vec{w}}$ points and negative on the opposite side.



$\frac{|w_0 + w_1 x'_1 + w_2 x'_2|}{\sqrt{w_1^2 + w_2^2}} = \frac{|\vec{w} \cdot \tilde{x}'|}{||\vec{w}||}$
$\vec{w} \cdot \tilde{x} = w_0 + w_1 x_1 + w_2 x_2$
$= 0$
$(x'_1, x'_2) = \tilde{x}'$
$(w_1, w_2) = \vec{w}$
$\frac{|w_0|}{||\vec{w}||} = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$
-2
-1
1
2
3

Margin

- Given a training set

$$D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_p, y_p)\}$$

Here $\vec{x}_k = (x_{k1} \ldots, x_{kn}) \in X \subseteq \mathbb{R}^n$ and $y_k \in \{-1, 1\}$.

Margin

- Given a training set

$$D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_p, y_p)\}$$

Here $\vec{x}_k = (x_{k1} \dots, x_{kn}) \in X \subseteq \mathbb{R}^n$ and $y_k \in \{-1, 1\}$.

- Assume that D is linearly separable, let $\vec{w}$ *be consistent with* D*.*

Margin of $\vec{w}$ is twice the minimum distance between feature vectors $\vec{x}_k$ and the separating hyperplane determined by $\vec{w}$, i.e.,

$$2 \min_k d[\vec{w}](\vec{x}_k) = 2 \min_k \frac{|\vec{w} \cdot \tilde{\mathbf{x}}_k|}{\|\underline{\vec{w}}\|}$$

- Our goal is to find $\vec{w}$ consistent with D that maximizes the margin.

Note that to maximize the margin it suffices to maximize $\min_k \frac{|\vec{w} \cdot \tilde{\mathbf{x}}_k|}{\|\underline{\vec{w}}\|}$ over $\vec{w}$ consistent with D.

Finding the Maximum Margin Classifier

We want to maximize the minimum distance of the feature vectors $\vec{x}_k$ from the separating hyperplane determined by $\vec{w}$.

Finding the Maximum Margin Classifier

We want to maximize the minimum distance of the feature vectors $\vec{x}_k$ from the separating hyperplane determined by $\vec{w}$.

Formally, we use the following:

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{|\vec{w} \cdot \tilde{\mathbf{x}}_k|}{||\underline{\vec{w}}||}$$

(= the distance of closest $\vec{x}_k$'s to the sep. hyperplane)

over the following constraints

$$\vec{w} \cdot \tilde{\mathbf{x}}_k > 0 \text{ for all } k \text{ satisfying } y_k = 1$$

$$\vec{w} \cdot \tilde{\mathbf{x}}_k < 0 \text{ for all } k \text{ satisfying } y_k = -1$$

(the contraints make sure that $\vec{w}$ is consistent with the training set D)

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{|\vec{w} \cdot \tilde{\mathbf{x}}_k|}{||\underline{\vec{w}}||}$$

over the following constraints

$\vec{w} \cdot \tilde{\mathbf{x}}_k > 0$ for all k satisfying $y_k = 1$

$\vec{w} \cdot \tilde{\mathbf{x}}_k < 0$ for all k satisfying $y_k = -1$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{|\vec{w} \cdot \tilde{\mathbf{x}}_k|}{||\underline{\vec{w}}||}$$

over the following constraints

$$\vec{w} \cdot \tilde{\mathbf{x}}_k > 0 \text{ for all } k \text{ satisfying } y_k = 1$$

$$\vec{w} \cdot \tilde{\mathbf{x}}_k < 0 \text{ for all } k \text{ satisfying } y_k = -1$$

Can be made more succinct:

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{||\underline{\vec{w}}||} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) > 0$$

The reason is that $\vec{w}$ is consistent with D. That is, $\vec{w} \cdot \tilde{x}_k > 0$ for $y_k = 1$, and $\vec{w} \cdot \tilde{x}_k < 0$ for $y_k = -1$.

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) > 0$$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) > 0$$

Observation: For every $\vec{w}$ satisfying $\min_k(y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) > 0$ there is $\vec{w}'$ satisfying $\min_k(y_k \cdot \vec{w}' \cdot \tilde{\mathbf{x}}_k) = 1$ such that

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} = \min_k \frac{y_k \cdot \vec{w}' \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}'\|}$$

Proof: Just consider $\vec{w}' = \vec{w}/\xi$ where $\xi = \min_k(y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k)$. □

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) > 0$$

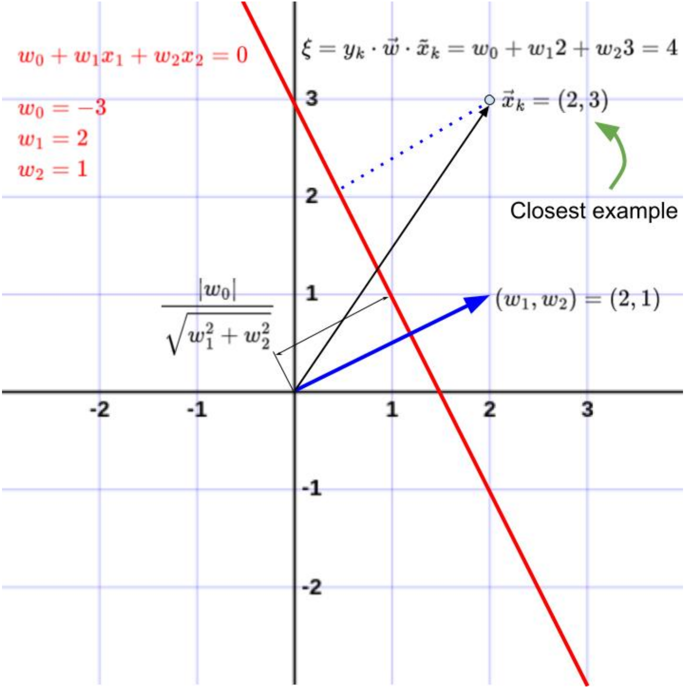
Observation: For every $\vec{w}$ satisfying $\min_k(y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) > 0$ there is $\vec{w}'$ satisfying $\min_k(y_k \cdot \vec{w}' \cdot \tilde{\mathbf{x}}_k) = 1$ such that

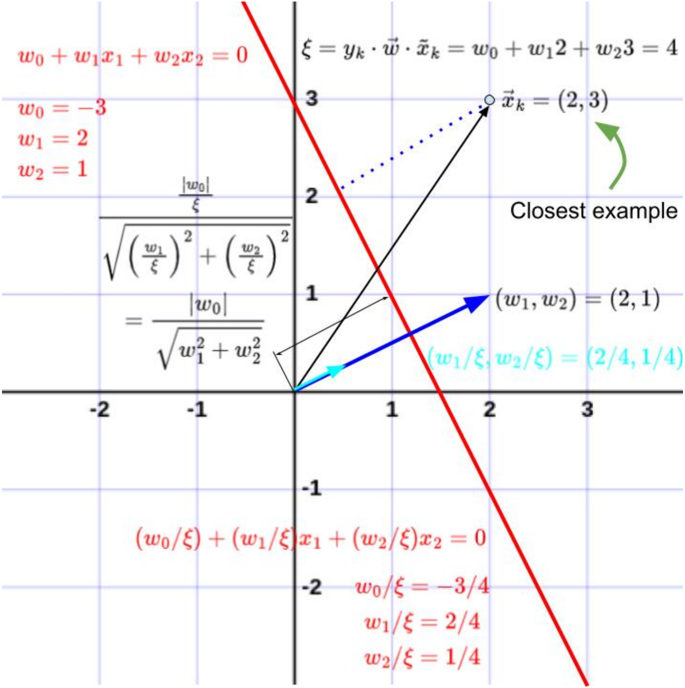
$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} = \min_k \frac{y_k \cdot \vec{w}' \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}'\|}$$

Proof: Just consider $\vec{w}' = \vec{w}/\xi$ where $\xi = \min_k(y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k)$. □

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$$


$w_0 + w_1 x_1 + w_2 x_2 = 0$
$w_0 = -3$
$w_1 = 2$
$w_2 = 1$
$\xi = y_k \cdot \vec{w} \cdot \tilde{x}_k = w_0 + w_1 2 + w_2 3 = 4$
$\vec{x}_k = (2, 3)$
Closest example
$(w_1, w_2) = (2, 1)$
$\frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$
3
2
1
-1
-2
-2
-1
1
2
3



$w_0 + w_1 x_1 + w_2 x_2 = 0$
$w_0 = -3$
$w_1 = 2$
$w_2 = 1$
$\xi = y_k \cdot \vec{w} \cdot \tilde{x}_k = w_0 + w_1 2 + w_2 3 = 4$
$\vec{x}_k = (2, 3)$
Closest example
$\frac{\frac{|w_0|}{\xi}}{\sqrt{\left(\frac{w_1}{\xi}\right)^2 + \left(\frac{w_2}{\xi}\right)^2}} = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$
$(w_1, w_2) = (2, 1)$
$(w_1/\xi, w_2/\xi) = (2/4, 1/4)$
$(w_0/\xi) + (w_1/\xi)x_1 + (w_2/\xi)x_2 = 0$
$w_0/\xi = -3/4$
$w_1/\xi = 2/4$
$w_2/\xi = 1/4$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\min_k \frac{y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$$

can be further simplified to

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{\|\vec{w}\|} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{||\underline{\vec{w}}||} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$$

Can be adjusted by loosening the constraints:

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{||\underline{\vec{w}}||} \quad \text{over} \quad \min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) \geq 1$$

If the latter is solved by $\vec{w}'$ with $\min_k (y_k \cdot \vec{w}' \cdot \tilde{\mathbf{x}}_k) > 1$, then

$$\min_k \frac{y_k \cdot \vec{w}' \cdot \tilde{\mathbf{x}}_k}{||\underline{\vec{w}'}||} > \frac{1}{||\underline{\vec{w}'}||} \geq \frac{1}{||\underline{\vec{w}}||} = \frac{\min_k y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k}{||\underline{\vec{w}}||}$$

For all $\vec{w}$ satisfying $\min_k (y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k) = 1$, which contradicts the fact that the maximum margin is attained by such a $\vec{w}$.

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1$$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{\|\underline{\vec{w}}\|} \quad \text{over} \quad \min_k y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1$$

Can be turned into

To maximize the margin, find $\vec{w}$ *minimizing*

$$||\underline{\vec{w}}|| \quad \text{over} \quad \min_k y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1$$

To maximize the margin, find $\vec{w}$ *maximizing*

$$\frac{1}{||\underline{\vec{w}}||} \quad \text{over} \quad \min_k y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1$$

Can be turned into

To maximize the margin, find $\vec{w}$ *minimizing*

$$||\underline{\vec{w}}|| \quad \text{over} \quad \min_k y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1$$

And, finally,

To maximize the margin, find $\vec{w}$ *minimizing*

$$\underline{\vec{w}} \cdot \underline{\vec{w}} \quad \text{over} \quad y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1 \text{ for all } k$$

Indeed, just note that $||\underline{\vec{w}}|| = \sqrt{\underline{\vec{w}} \cdot \underline{\vec{w}}}$.

SVM – Optimization

Assume a given training set

$$D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_p, y_p)\}$$

Here $\vec{x}_k = (x_{k1} \ldots, x_{kn}) \in X \subseteq \mathbb{R}^n$ and $y_k \in \{-1, 1\}$.
(recall $\tilde{x}_k = (x_{k0}, x_{k1}, \ldots, x_{kn})$ where $x_{k0} = 1$)

Margin maximization as a *quadratic optimization problem:*

Find $\vec{w}$ *minimizing*

$$\underline{\vec{w}} \cdot \underline{\vec{w}}$$

under the constraints

$$y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1 \text{ for all } k$$

Support vectors are vectors $\vec{x}_k$ closest to the *optimal* separating hyperplane, i.e., those satisfying $y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k = 1$ for a minimizing $\vec{w}$.

Example

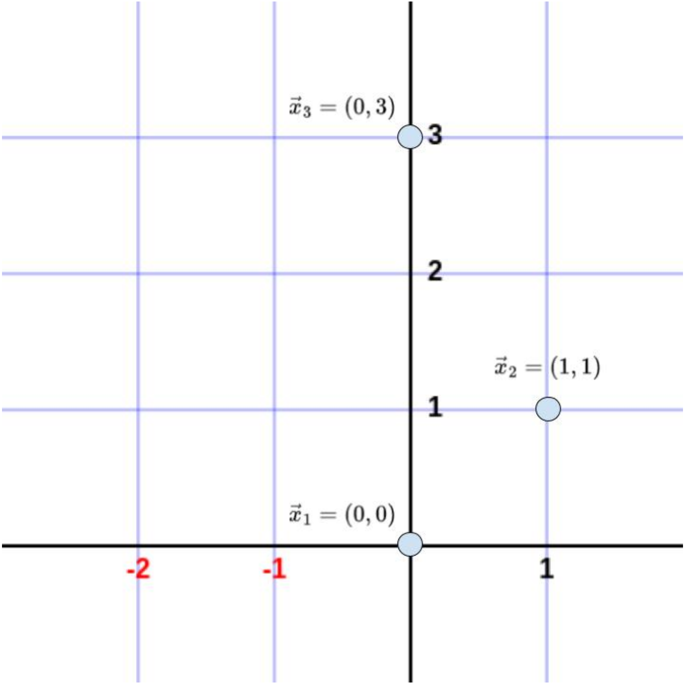
Training set:

$$D = \{((0,0),-1),((1,1),1),((0,3),1)\}$$

That is

$$\begin{aligned} \vec{x}_1 &= (0,0) \\ \vec{x}_2 &= (1,1) \\ \vec{x}_3 &= (0,3) \end{aligned} \qquad \begin{aligned} \tilde{\mathbf{x}}_1 &= (1,0,0) \\ \tilde{\mathbf{x}}_2 &= (1,1,1) \\ \tilde{\mathbf{x}}_3 &= (1,0,3) \end{aligned}$$

$$\begin{aligned} y_1 &= -1 \\ y_2 &= 1 \\ y_3 &= 1 \end{aligned}$$



$\vec{x}_3 = (0, 3)$
3
2
$\vec{x}_2 = (1, 1)$
1
$\vec{x}_1 = (0, 0)$
-2
-1
1

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$(-1) \cdot (1w_0 + 0w_1 + 0w_2) = -w_0 \geq 1$$
$$1 \cdot (1w_0 + 1w_1 + 1w_2) = w_0 + w_1 + w_2 \geq 1$$
$$1 \cdot (1w_0 + 0w_1 + 3w_2) = w_0 + 3w_2 \geq 1$$

It can be solved using a quadratic programming solver.

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$(-1) \cdot (1w_0 + 0w_1 + 0w_2) = -w_0 \geq 1$$
$$1 \cdot (1w_0 + 1w_1 + 1w_2) = w_0 + w_1 + w_2 \geq 1$$
$$1 \cdot (1w_0 + 0w_1 + 3w_2) = w_0 + 3w_2 \geq 1$$

It can be solved using a quadratic programming solver.

To solve by hand, assume that we know that $\vec{x}_1$ and $\vec{x}_2$ are support vectors.

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$-w_0 = 1$$
$$w_0 + w_1 + w_2 = 1$$
$$w_0 + 3w_2 \geq 1$$

Note that the equality constraints correspond to our assumption that $\vec{x}_1$ and $\vec{x}_2$ are support vectors.

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$\begin{aligned} -w_0 &= 1 \\ w_0 + w_1 + w_2 &= 1 \\ w_0 + 3w_2 &\geq 1 \end{aligned}$$

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$\begin{aligned} -w_0 &= 1 \\ w_0 + w_1 + w_2 &= 1 \\ w_0 + 3w_2 &\geq 1 \end{aligned}$$

Can be transformed to

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$\begin{aligned} w_1 + w_2 &= 2 \\ 3w_2 &\geq 2 \end{aligned}$$

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$\begin{aligned} w_1 + w_2 &= 2 \\ 3w_2 &\geq 2 \end{aligned}$$

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$\begin{aligned} w_1 + w_2 &= 2 \\ 3w_2 &\geq 2 \end{aligned}$$

Substituting $w_2 = 2 - w_1$ into the quadratic function we obtain

$$w_1^2 + (2 - w_1)^2 = w_1^2 + w_1^2 - 4w_1 + 4 = 2w_1^2 - 4w_1 + 4$$

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$w_1 + w_2 = 2$$
$$3w_2 \geq 2$$

Substituting $w_2 = 2 - w_1$ into the quadratic function we obtain

$$w_1^2 + (2 - w_1)^2 = w_1^2 + w_1^2 - 4w_1 + 4 = 2w_1^2 - 4w_1 + 4$$

substituting $w_2 = 2 - w_1$ into the inequality $3w_2 \geq 2$ we obtain

$$6 - 3w_1 \geq 2$$

Find $\vec{w}$ minimizing $w_1^2 + w_2^2$ under the constraints

$$w_1 + w_2 = 2$$
$$3w_2 \geq 2$$

Substituting $w_2 = 2 - w_1$ into the quadratic function we obtain

$$w_1^2 + (2 - w_1)^2 = w_1^2 + w_1^2 - 4w_1 + 4 = 2w_1^2 - 4w_1 + 4$$

substituting $w_2 = 2 - w_1$ into the inequality $3w_2 \geq 2$ we obtain

$$6 - 3w_1 \geq 2$$

This reduces our problem to

Find $\vec{w}$ minimizing $2w_1^2 - 4w_1 + 4$ under the constraint $w_1 \leq \frac{4}{3}$

Find $\vec{w}$ minimizing $2w_1^2 - 4w_1 + 4$ under the constraint $w_1 \leq \frac{4}{3}$

Find $\vec{w}$ minimizing $2w_1^2 - 4w_1 + 4$ under the constraint $w_1 \leq \frac{4}{3}$

Is solved by

$$w_1 = 1$$

Find $\vec{w}$ minimizing $2w_1^2 - 4w_1 + 4$ under the constraint $w_1 \leq \frac{4}{3}$

Is solved by

$$w_1 = 1$$

From $w_2 = 2 - w_1$ we obtain

$$w_2 = 2 - 1 = 1$$

Find $\vec{w}$ minimizing $2w_1^2 - 4w_1 + 4$ under the constraint $w_1 \leq \frac{4}{3}$

Is solved by

$$w_1 = 1$$

From $w_2 = 2 - w_1$ we obtain

$$w_2 = 2 - 1 = 1$$

From $-w_0 = 1$ we obtain

$$w_0 = -1$$

Find $\vec{w}$ minimizing $2w_1^2 - 4w_1 + 4$ under the constraint $w_1 \leq \frac{4}{3}$

Is solved by

$$w_1 = 1$$

From $w_2 = 2 - w_1$ we obtain

$$w_2 = 2 - 1 = 1$$

From $-w_0 = 1$ we obtain

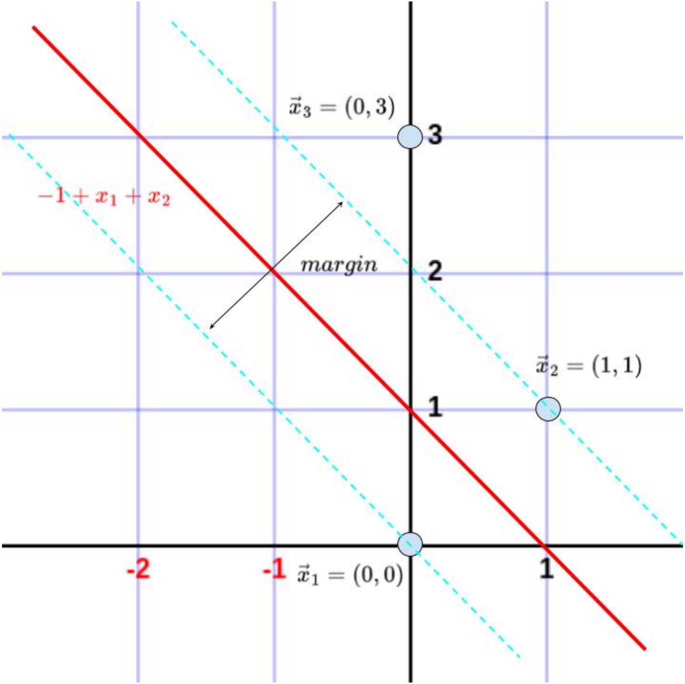
$$w_0 = -1$$

The final model is

$$h[\vec{w}](\vec{x}) = -1 + x_1 + x_2$$

The separating hyperplane is determined by

$$-1 + x_1 + x_2 = 0$$



$\vec{x}_3 = (0,3)$
3
$-1 + x_1 + x_2$
margin
2
$\vec{x}_2 = (1,1)$
1
-2
-1
$\vec{x}_1 = (0,0)$
1

SVM – Optimization

- Need to optimize a quadratic function subject to linear constraints.

SVM – Optimization

- Need to optimize a quadratic function subject to linear constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which efficient methods (and tools) exist.

But why has the SVM been so successful?

... the improvement by finding the maximum margin classifier does not seem to be so strong ... right?

SVM – Optimization

- Need to optimize a quadratic function subject to linear constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which efficient methods (and tools) exist.

But why has the SVM been so successful?

... the improvement by finding the maximum margin classifier does not seem to be so strong ... right?

The answer lies in their ability to deal with non-linearly separable sets efficiently using the *kernel trick* (see a later lecture).

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.
- For large problems, this is usually not possible; many methods avoiding direct solutions have been devised.

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.
- For large problems, this is usually not possible; many methods avoiding direct solutions have been devised.
- These methods usually decompose the optimization problem into a sequence of smaller ones. Intuitively,

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.
- For large problems, this is usually not possible; many methods avoiding direct solutions have been devised.
- These methods usually decompose the optimization problem into a sequence of smaller ones. Intuitively,
  - starts with a (smaller) subset of training examples.

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.
- For large problems, this is usually not possible; many methods avoiding direct solutions have been devised.
- These methods usually decompose the optimization problem into a sequence of smaller ones. Intuitively,
  - starts with a (smaller) subset of training examples.
  - Find an optimal solution using any solver.

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.
- For large problems, this is usually not possible; many methods avoiding direct solutions have been devised.
- These methods usually decompose the optimization problem into a sequence of smaller ones. Intuitively,
  - starts with a (smaller) subset of training examples.
  - Find an optimal solution using any solver.
  - Afterwards, only support vectors matter in the solution! Leave only them in the training set, and add new training examples.

Comments on Algorithms

- The main bottleneck of SVM is quadratic programming (QP) complexity. A naive QP solver has cubic complexity.
- For small problems, any general-purpose optimization algorithm can be used.
- For large problems, this is usually not possible; many methods avoiding direct solutions have been devised.
- These methods usually decompose the optimization problem into a sequence of smaller ones. Intuitively,
    - starts with a (smaller) subset of training examples.
    - Find an optimal solution using any solver.
    - Afterwards, only support vectors matter in the solution! Leave only them in the training set, and add new training examples.
    - This iterative procedure decreases the (general) cost function.

Soft-margin SVM

Trade-off few misclassifications with a wide margin for the rest.

Find $\vec{w}$ *minimizing*

$$\underline{\vec{w}} \cdot \underline{\vec{w}} + C \sum_k \zeta_k$$

C is a hyperparameter

under the constraints

$$y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1 - \zeta_k \text{ for all } k$$

$$\zeta_k \geq 0 \text{ for all } k$$

Soft-margin SVM

Trade-off few misclassifications with a wide margin for the rest.

Find $\vec{w}$ *minimizing*

$$\underline{\vec{w}} \cdot \underline{\vec{w}} + C \sum_k \zeta_k \qquad C \text{ is a hyperparameter}$$

under the constraints

$$y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k \geq 1 - \zeta_k \text{ for all } k$$

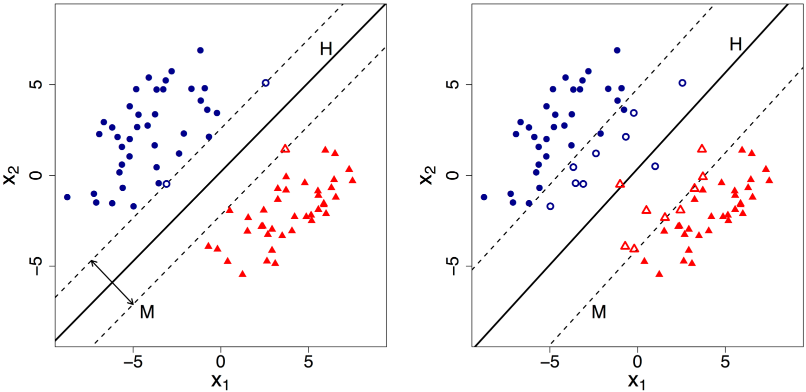
$$\zeta_k \geq 0 \text{ for all } k$$

Which is the same as the following *unconstrained* optimization:

Find $\vec{w}$ *minimizing* the *hinge loss*

$$\underline{\vec{w}} \cdot \underline{\vec{w}} + C \sum_k \max(0, 1 - y_k \cdot \vec{w} \cdot \tilde{\mathbf{x}}_k)$$

Hard vs Soft Margin SVM



Source: Dishaa Agarwal

https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/

Comments on SVM

- SVMs were originally proposed by Boser, Guyon, and Vapnik in 1992, and gained increasing popularity in the late 1990s.

Comments on SVM

- SVMs were originally proposed by Boser, Guyon, and Vapnik in 1992, and gained increasing popularity in the late 1990s.
- SVMs are currently among the best performers for several classification tasks ranging from text to genomic data.

Comments on SVM

- SVMs were originally proposed by Boser, Guyon, and Vapnik in 1992, and gained increasing popularity in the late 1990s.
- SVMs are currently among the best performers for several classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g., graphs, sequences, relational data) by designing kernel functions for such data.

Comments on SVM

- SVMs were originally proposed by Boser, Guyon, and Vapnik in 1992, and gained increasing popularity in the late 1990s.
- SVMs are currently among the best performers for several classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g., graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to several tasks, such as regression [Vapnik et al. '97], principal component analysis [Schölkopf et al. '99], etc.