Learning Maximum-Margin Hyperplanes: Support Vector Machines

soft margin

hard margin

hands on
session

Piyush Rai

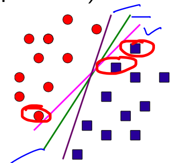
Machine Learning (CS771A)

Aug 24, 2016

Perceptron and (Lack of) Margins

- Perceptron learns a hyperplane (of many possible) that separates the classes

2 - dim



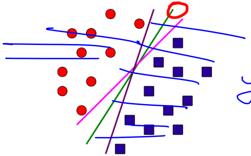
more preferred

green - max degree of sep

Perceptron and (Lack of) Margins

- Perceptron learns a hyperplane (of many possible) that separates the classes

+1



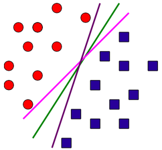
$y = -1$

0 Training Error

- Standard Perceptron doesn't guarantee any "margin" around the hyperplane

Perceptron and (Lack of) Margins

- Perceptron learns a hyperplane (of many possible) that separates the classes



- Standard Perceptron doesn't guarantee any "margin" around the hyperplane
- Note: Possible to "artificially" introduce a margin in the Perceptron

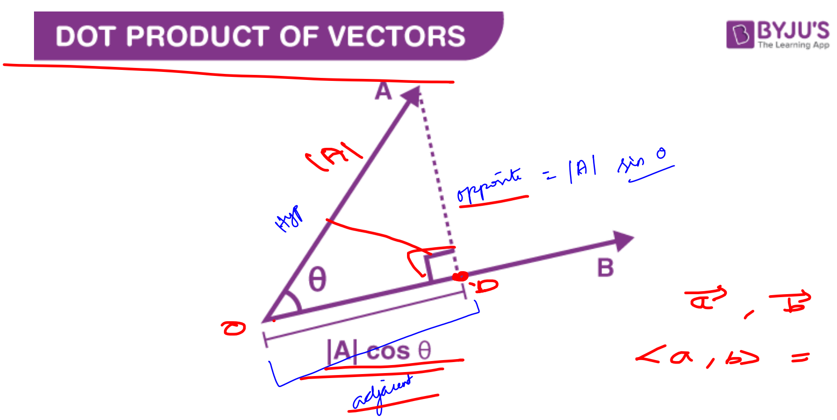
$$\vec{a} = [1, 2]$$

$$\vec{b} = [2, 3]$$

$$\vec{a} \cdot \vec{b} = \sum_i a_i b_i = 2 + 6 = 8$$

$$= \|\vec{a}\| \, \|\vec{b}\| \cos\theta$$

dot Products and margin



$\sin\theta = \frac{opp}{hyp}$

$= \frac{OD}{OA}$

$\cos\theta = \frac{OD}{OA}$

$\vec{a}, \vec{b}$

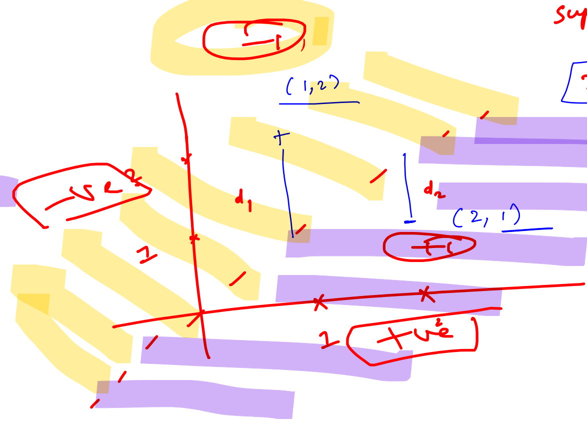
$\langle a, b \rangle =$

suppose Hyp plane is

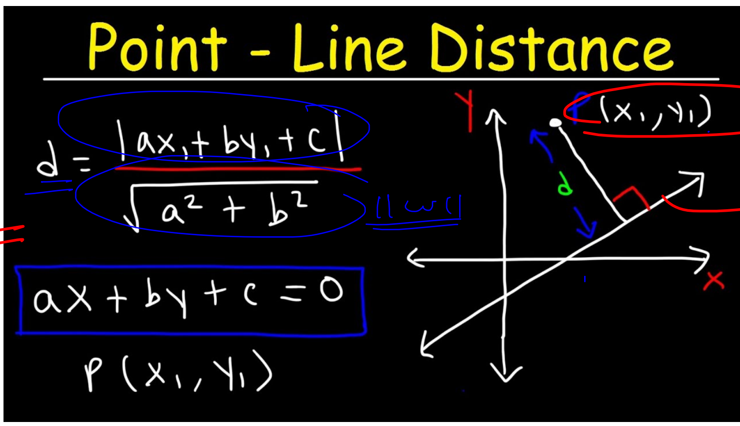
$\boxed{x = y}$

Semantic of

Sign $(w^T x)$



can we compute the distance b/w point and the hyper plane



$ax + by + c$

Hyper plane is $\boxed{x - y = 0}$

$a = 1$ and $b = -1$ $c = 0$

d1 d2

$d_1 = \frac{|1 - 2 + 0|}{\sqrt{1+4}} = \frac{-1}{\sqrt{5}}$

$d_2 = \frac{|2 - 1 + 0|}{\sqrt{5}} = \frac{+1}{\sqrt{5}}$

dot Product b/w line and point

denote $\vec{w}$ as coefficients of the line

$$\vec{w} = \begin{array}{ccc} x & y & c \\ \left[1, \right. & -1, & \left. 0 \right] \\ a & b & c \end{array}$$

$P_1 = (1, 2)$

$P_2 = (2, 1)$

augment each point with 1 towards the end to account for 'c'

$x_1 = [1, 2, 1]$ and $x_2 = [2, 1, 1]$

$w^T x_1 = 1 - 2 + 0 = -1$ $\frac{}{\sqrt{5}}$ and $w^T x_2 = 2 - 1 + 0 = 1$ $\frac{}{\sqrt{5}}$

This is unnormalized signed distance.

[1, 2]

Norm factor $= \frac{1}{\|w\|} = \frac{1}{\sqrt{1^2 + 2^2 + 0^2}} = \frac{1}{\sqrt{5}}$

Sign Indicates the side of the margin

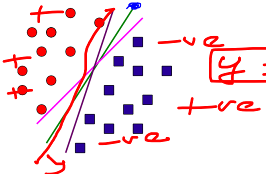
Perceptron and (Lack of) Margins

- Perceptron learns a hyperplane (of many possible) that separates the classes

$\boxed{y\,(w^T x + b)} > 0$

$x \in$ +ve class

$y = +1$



$\boxed{y = -1}$

dot product

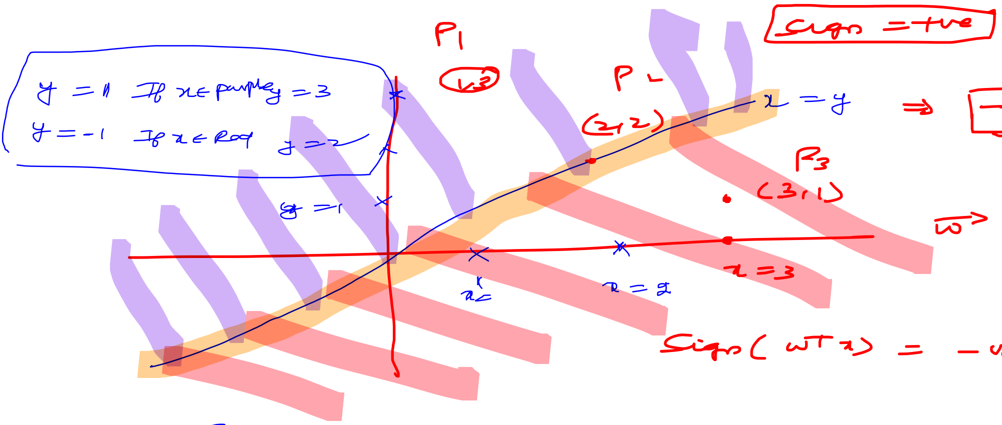
- Standard Perceptron doesn't guarantee any "margin" around the hyperplane
- Note: Possible to "artificially" introduce a margin in the Perceptron
  - Simply change the Perceptron mistake condition to

$$y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \leq \gamma$$

where $\gamma > 0$ is a pre-specified margin. For standard Perceptron, $\gamma = 0$

$\boxed{y\,(w^T x + b) > \gamma}$

$\forall\, x \in$ Dataset



$\Rightarrow \boxed{-x + y = 0}$

$\vec{w} = [-1, 1, 0]$

$\text{sign}(w^T x) = -ve$

For pt P_1

$$w^T P_1 = -1 + 3 + 0 = +2$$

For pt P_2

$$w^T P_2 = -2 + 2 = 0$$

$\{ \; w^T P_3 = -2$

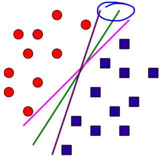
Solve for a $\vec{w}$ such that

$$\boxed{y\,(w^T x + b) \geq \, ?} \quad \forall x \in \text{Trn data}$$

$(w^T x + b)$: unnormalized distance

Perceptron and (Lack of) Margins

- Perceptron learns a hyperplane (of many possible) that separates the classes



$\gamma \geq 0$

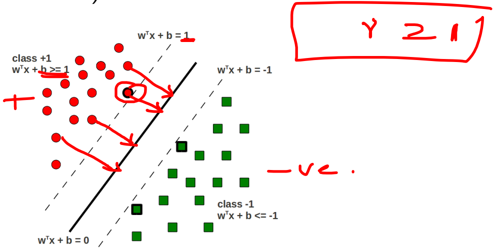
- Standard Perceptron doesn't guarantee any "margin" around the hyperplane
- Note: Possible to "artificially" introduce a margin in the Perceptron
    - Simply change the Perceptron mistake condition to

$$y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \leq \gamma$$

    where $\gamma > 0$ is a pre-specified margin. For standard Perceptron, $\gamma = 0$
    - **Support Vector Machine (SVM)** offers a more principled way of doing this by learning the maximum margin hyperplane

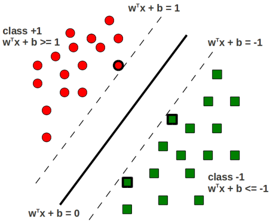
Support Vector Machine (SVM)

- Learns a hyperplane such that the positive and negative class training examples are as far away as possible from it (ensures good generalization)



Support Vector Machine (SVM)

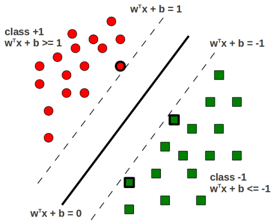
- Learns a hyperplane such that the positive and negative class training examples are as far away as possible from it (ensures good generalization)



- SVMs can also learn nonlinear decision boundaries using **kernels** (though the idea of kernels is not specific to SVMs and is more generally applicable)

Support Vector Machine (SVM)

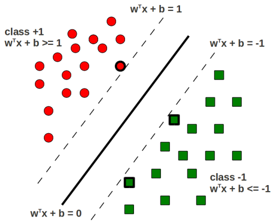
- Learns a hyperplane such that the positive and negative class training examples are as far away as possible from it (ensures good generalization)



- SVMs can also learn nonlinear decision boundaries using **kernels** (though the idea of kernels is not specific to SVMs and is more generally applicable)
- Reason behind the name "Support Vector Machine"?

Support Vector Machine (SVM)

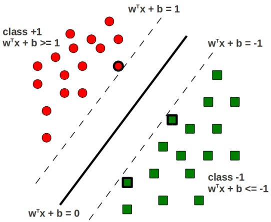
- Learns a hyperplane such that the positive and negative class training examples are as far away as possible from it (ensures good generalization)



- SVMs can also learn nonlinear decision boundaries using **kernels** (though the idea of kernels is not specific to SVMs and is more generally applicable)
- Reason behind the name "Support Vector Machine"? SVM finds the most important examples (called "support vectors") in the training data

Support Vector Machine (SVM)

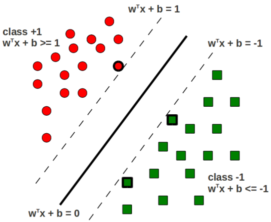
- Learns a hyperplane such that the positive and negative class training examples are as far away as possible from it (ensures good generalization)



- SVMs can also learn nonlinear decision boundaries using **kernels** (though the idea of kernels is not specific to SVMs and is more generally applicable)
- Reason behind the name "Support Vector Machine"? SVM finds the most important examples (called "support vectors") in the training data
  - These examples also "balance" the margin boundaries (hence called "support").

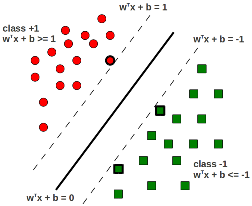
Support Vector Machine (SVM)

- Learns a hyperplane such that the positive and negative class training examples are as far away as possible from it (ensures good generalization)



- SVMs can also learn nonlinear decision boundaries using **kernels** (though the idea of kernels is not specific to SVMs and is more generally applicable)
- Reason behind the name "Support Vector Machine"? SVM finds the most important examples (called "support vectors") in the training data
  - These examples also "balance" the margin boundaries (hence called "support"). Also, even if we throw away the remaining training data and re-learn the SVM classifier, we'll get the same hyperplane

Learning a Maximum Margin Hyperplane

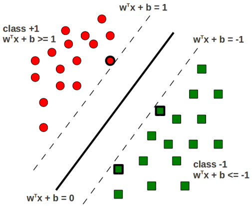


$(\vec{w}, b)$

- *Suppose* there exists a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ such that
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \geq 1$ for $y_n = +1$
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \leq -1$ for $y_n = -1$

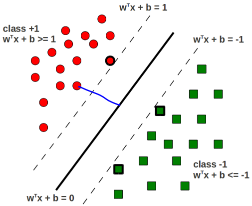
$\boxed{y_n(w^T x_n + b) \geq 1}$

Learning a Maximum Margin Hyperplane



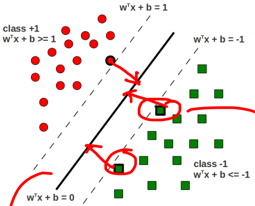
- *Suppose* there exists a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ such that
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \geq 1$ for $y_n = +1$
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \leq -1$ for $y_n = -1$
  - Equivalently, $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 \quad \forall n$ **(the margin condition)**

Learning a Maximum Margin Hyperplane



- *Suppose* there exists a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ such that
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \geq 1$ for $y_n = +1$
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \leq -1$ for $y_n = -1$
  - Equivalently, $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 \quad \forall n$ **(the margin condition)**
  - Also note that $\min_{1 \leq n \leq N} |\boldsymbol{w}^T \boldsymbol{x}_n + b| = 1$

Learning a Maximum Margin Hyperplane



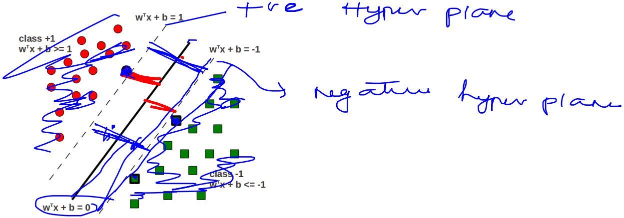
$|w^T x + b|$ = unnorm distance

- *Suppose* there exists a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ such that
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \geq 1$ for $y_n = +1$
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \leq -1$ for $y_n = -1$
  - Equivalently, $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 \quad \forall n$ **(the margin condition)**
  - Also note that $\min_{1 \leq n \leq N} |\boldsymbol{w}^T \boldsymbol{x}_n + b| = 1$
  - Thus margin on each side: $\gamma = \min_{1 \leq n \leq N} \frac{|\boldsymbol{w}^T \boldsymbol{x}_n + b|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$

$\max \frac{1}{||w||}$

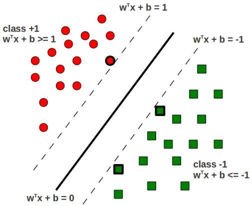
s.t. $y_n(w^T x_n + b) \geq 1 \quad \forall n$

Learning a Maximum Margin Hyperplane



- *Suppose* there exists a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ such that
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \geq 1$ for $y_n = +1$
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \leq -1$ for $y_n = -1$
  - Equivalently, $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 \quad \forall n$ **(the margin condition)**
  - Also note that $\min_{1 \leq n \leq N} |\boldsymbol{w}^T \boldsymbol{x}_n + b| = 1$
  - Thus margin on each side: $\gamma = \min_{1 \leq n \leq N} \frac{|\boldsymbol{w}^T \boldsymbol{x}_n + b|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$
  - Total margin $= 2\gamma = \frac{2}{||\boldsymbol{w}||}$

Learning a Maximum Margin Hyperplane



- *Suppose* there exists a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ such that
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \geq 1$ for $y_n = +1$
  - $\boldsymbol{w}^T \boldsymbol{x}_n + b \leq -1$ for $y_n = -1$
  - Equivalently, $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 \quad \forall n$ **(the margin condition)**
  - Also note that $\min_{1 \leq n \leq N} |\boldsymbol{w}^T \boldsymbol{x}_n + b| = 1$
  - Thus margin on each side: $\gamma = \min_{1 \leq n \leq N} \frac{|\boldsymbol{w}^T \boldsymbol{x}_n + b|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$
  - Total margin $= 2\gamma = \frac{2}{||\boldsymbol{w}||}$
- Want the hyperplane $(\boldsymbol{w}, b)$ to have the largest possible margin

Large Margin = Good Generalization

- Large margins intuitively mean good generalization

Large Margin = Good Generalization

- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$

Large Margin = Good Generalization

- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
- Large margin $\Rightarrow$ small $||\boldsymbol{w}||$, i.e., small ℓ_2 norm of $\boldsymbol{w}$

Large Margin = Good Generalization

- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
- Large margin $\Rightarrow$ small $||\boldsymbol{w}||$, i.e., small ℓ_2 norm of $\boldsymbol{w}$
- Small $||\boldsymbol{w}|| \Rightarrow$ regularized/simple solutions (w_i's don't become too large)

Large Margin = Good Generalization

- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
- Large margin $\Rightarrow$ small $||\boldsymbol{w}||$, i.e., small ℓ_2 norm of $\boldsymbol{w}$
- Small $||\boldsymbol{w}|| \Rightarrow$ regularized/simple solutions (w_i's don't become too large)
  - Recall our discussion of regularization..

Large Margin = Good Generalization

- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
- Large margin $\Rightarrow$ small $||\boldsymbol{w}||$, i.e., small ℓ_2 norm of $\boldsymbol{w}$
- Small $||\boldsymbol{w}|| \Rightarrow$ regularized/simple solutions (w_i's don't become too large)
  - Recall our discussion of regularization..
- Simple solutions $\Rightarrow$ good generalization on test data

Large Margin = Good Generalization

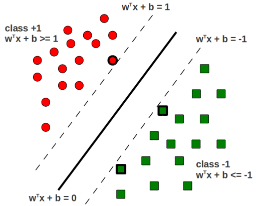
- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
- Large margin $\Rightarrow$ small $||\boldsymbol{w}||$, i.e., small ℓ_2 norm of $\boldsymbol{w}$
- Small $||\boldsymbol{w}|| \Rightarrow$ regularized/simple solutions (w_i's don't become too large)
    - Recall our discussion of regularization..
- Simple solutions $\Rightarrow$ good generalization on test data
- Want to see an even more formal justification? :-)

Large Margin = Good Generalization

- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
- Large margin $\Rightarrow$ small $||\boldsymbol{w}||$, i.e., small ℓ_2 norm of $\boldsymbol{w}$
- Small $||\boldsymbol{w}|| \Rightarrow$ regularized/simple solutions (w_i's don't become too large)
    - Recall our discussion of regularization..
- Simple solutions $\Rightarrow$ good generalization on test data
- Want to see an even more formal justification? :-)
    - Wait until we cover Learning Theory!

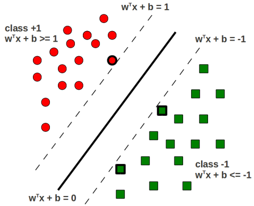
Hard-Margin SVM

- Every training example has to fulfil the margin condition $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1$



Hard-Margin SVM

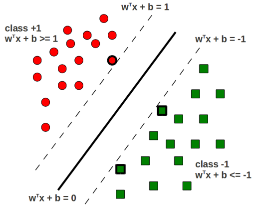
- Every training example has to fulfil the margin condition $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1$



- Also want to maximize the margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$

Hard-Margin SVM

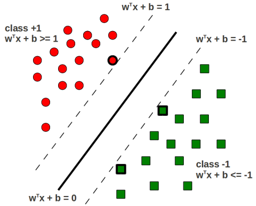
- Every training example has to fulfil the margin condition $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1$



- Also want to maximize the margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
  - Equivalent to minimizing $||\boldsymbol{w}||^2$ or $\frac{||\boldsymbol{w}||^2}{2}$

Hard-Margin SVM

- Every training example has to fulfil the margin condition $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1$



- Also want to maximize the margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
  - Equivalent to minimizing $||\boldsymbol{w}||^2$ or $\frac{||\boldsymbol{w}||^2}{2}$
- The objective for hard-margin SVM

opt pblm to be solved

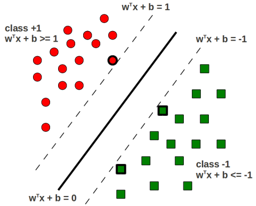
$$\boxed{\begin{aligned} &\min_{\boldsymbol{w},b} \quad f(\boldsymbol{w}, b) = \frac{||\boldsymbol{w}||^2}{2} \\ &\text{subject to} \quad y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1, \qquad n = 1, \ldots, N \end{aligned}}$$

→ quadratic cost function

→ linear constraint function

Hard-Margin SVM

- Every training example has to fulfil the margin condition $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1$



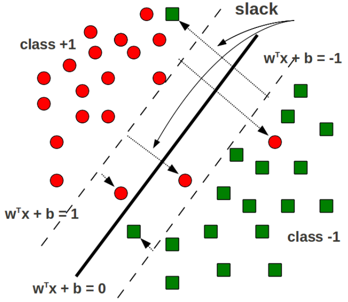
- Also want to maximize the margin $\gamma \propto \frac{1}{||\boldsymbol{w}||}$
  - Equivalent to minimizing $||\boldsymbol{w}||^2$ or $\frac{||\boldsymbol{w}||^2}{2}$
- The objective for hard-margin SVM

$$\boxed{\begin{array}{ll} \min_{\boldsymbol{w},b} & f(\boldsymbol{w},b) = \frac{||\boldsymbol{w}||^2}{2} \\ \text{subject to} & y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1, \qquad n = 1,\ldots,N \end{array}}$$

- Thus the hard-margin SVM minimizes a convex objective function which is a Quadratic Program (QP) with N linear inequality constraints

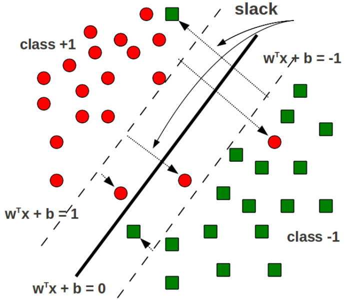
Soft-Margin SVM (More Commonly Used)

- Allow some training examples to fall **within** the margin region, or be even misclassified (i.e., fall on the wrong side). Preferable if training data is noisy



Soft-Margin SVM (More Commonly Used)

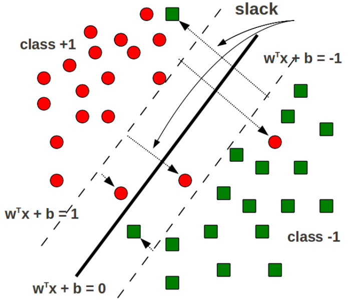
- Allow some training examples to fall **within** the margin region, or be even misclassified (i.e., fall on the wrong side). Preferable if training data is noisy



- Each training example $(\boldsymbol{x}_n, y_n)$ given a "slack" $\xi_n \geq 0$ (distance by which it "violates" the margin). If $\xi_n > 1$ then $\boldsymbol{x}_n$ is totally on the wrong side

Soft-Margin SVM (More Commonly Used)

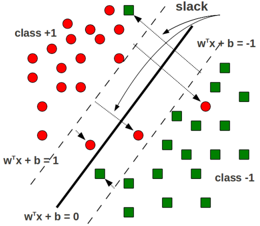
- Allow some training examples to fall **within** the margin region, or be even misclassified (i.e., fall on the wrong side). Preferable if training data is noisy



- Each training example $(\boldsymbol{x}_n, y_n)$ given a "slack" $\xi_n \geq 0$ (distance by which it "violates" the margin). If $\xi_n > 1$ then $\boldsymbol{x}_n$ is totally on the wrong side
  - Basically, we want a soft-margin condition: $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0$

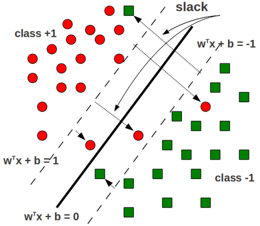
Soft-Margin SVM (More Commonly Used)

- Goal: Maximize the margin, while also minimizing the sum of slacks (don't want too many training examples violating the margin condition)



Soft-Margin SVM (More Commonly Used)

- Goal: Maximize the margin, while also minimizing the sum of slacks (don't want too many training examples violating the margin condition)



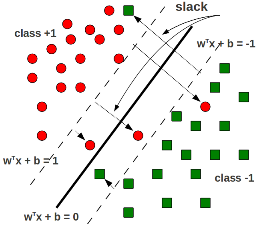
- The primal objective for soft-margin SVM can thus be written as

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad f(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{||\boldsymbol{w}||^2}{2} + C \sum_{n=1}^{N} \xi_n$$

$$\text{subject to constraints} \quad y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \qquad n = 1, \ldots, N$$

Soft-Margin SVM (More Commonly Used)

- Goal: Maximize the margin, while also minimizing the sum of slacks (don't want too many training examples violating the margin condition)



- The primal objective for soft-margin SVM can thus be written as

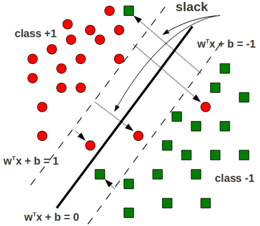
$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad f(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{||\boldsymbol{w}||^2}{2} + C\sum_{n=1}^{N} \xi_n$$

$$\text{subject to constraints} \quad y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \qquad n = 1, \ldots, N$$

- Thus the soft-margin SVM also minimizes a convex objective function which is a Quadratic Program (QP) with $2N$ linear inequality constraints

Soft-Margin SVM (More Commonly Used)

- Goal: Maximize the margin, while also minimizing the sum of slacks (don't want too many training examples violating the margin condition)



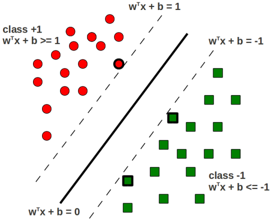
- The primal objective for soft-margin SVM can thus be written as

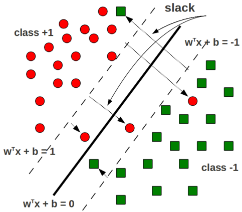
$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad f(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{||\boldsymbol{w}||^2}{2} + C\sum_{n=1}^{N}\xi_n$$

subject to constraints $\quad y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad n = 1, \ldots, N$

- Thus the soft-margin SVM also minimizes a convex objective function which is a Quadratic Program (QP) with $2N$ linear inequality constraints
- Param. C controls the trade-off between large margin vs small training error

Summary: Hard-Margin SVM vs Soft-Margin SVM





- Objective for the hard-margin SVM (unknowns are $\boldsymbol{w}$ and b)

$$\min_{\boldsymbol{w},b} \quad f(\boldsymbol{w},b) = \frac{||\boldsymbol{w}||^2}{2}$$

$$\text{subject to constraints} \quad y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1, \qquad n = 1,\ldots,N$$

- Objective for the soft-margin SVM (unknowns are $\boldsymbol{w}$, b, and $\{\xi_n\}_{n=1}^N$)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad f(\boldsymbol{w},b,\boldsymbol{\xi}) = \frac{||\boldsymbol{w}||^2}{2} + C\sum_{n=1}^{N}\xi_n$$

$$\text{subject to} \quad y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \qquad n = 1,\ldots,N$$

- In either case, we have to solve constrained, convex optimization problem

Brief Detour: Solving Constrained Optimization Problems

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$
\begin{aligned}
&\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \\
&\text{s.t} \quad g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N \\
&\qquad \cancel{h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M}
\end{aligned}
$$

let us Ignore the Equality constraints since SVM

does not have them

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$
\begin{aligned}
&\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \\
&\text{s.t} \quad g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N \\
&\qquad \cancel{h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M}
\end{aligned}
$$

- Introduce Lagrange multipliers $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^N$, $\alpha_n \geq 0$, and $\cancel{\boldsymbol{\beta} = \{\beta_m\}_{m=1}^M}$, one for each constraint, and construct the following Lagrangian

$$
\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g_n(\boldsymbol{w}) + \cancel{\sum_{m=1}^{N} \beta_n h_n(\boldsymbol{w})}
$$

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\begin{aligned}
&\min_{\boldsymbol{w}} \ f(\boldsymbol{w}) \\
\text{s.t} \quad & g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N \\
& h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M
\end{aligned}$$

- Introduce Lagrange multipliers $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^{N}$, $\alpha_n \geq 0$, and $\boldsymbol{\beta} = \{\beta_m\}_{m=1}^{M}$, one for each constraint, and construct the following Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g_n(\boldsymbol{w}) + \sum_{m=1}^{N} \cancel{\beta_n h_n}(\boldsymbol{w})$$

- Consider $\mathcal{L}_P(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Note that

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\min_{\boldsymbol{w}} \; f(\boldsymbol{w})$$
$$\text{s.t} \quad g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N$$
$$h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M$$

- Introduce Lagrange multipliers $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^N$, $\alpha_n \geq 0$, and $\boldsymbol{\beta} = \{\beta_m\}_{m=1}^M$, one for each constraint, and construct the following Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g_n(\boldsymbol{w}) + \sum_{m=1}^{N} \beta_n h_n(\boldsymbol{w})$$

- Consider $\mathcal{L}_P(\boldsymbol{w}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Note that
  - $\mathcal{L}_P(\boldsymbol{w}) = \infty$ if $\boldsymbol{w}$ violates any of the constraints (g's or h's)

assume $g_n(w)$ is violated

$\Rightarrow \quad g_n(w) > 0$

Now when we max$_{\alpha_n}$ $\alpha_n g_n(w)$ we can trivially

set $\alpha_n = +\infty$ and the obj is maximized

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\min_{\boldsymbol{w}} \; f(\boldsymbol{w})$$
$$\text{s.t} \quad g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N$$
$$h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M$$

- Introduce Lagrange multipliers $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^N$, $\alpha_n \geq 0$, and $\boldsymbol{\beta} = \{\beta_m\}_{m=1}^M$, one for each constraint, and construct the following Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{n=1}^N \alpha_n g_n(\boldsymbol{w}) + \sum_{m=1}^N \beta_n h_n(\boldsymbol{w})$$

- Consider $\mathcal{L}_P(\boldsymbol{w}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Note that
  - $\mathcal{L}_P(\boldsymbol{w}) = \infty$ if $\boldsymbol{w}$ violates any of the constraints (g's or h's) If $g_n(w)$ is satisfied
  - $\mathcal{L}_P(\boldsymbol{w}) = f(\boldsymbol{w})$ if $\boldsymbol{w}$ satisfies all the constraints (g's and h's)

$g_n(w) \leq 0$

what is α_n in $\max_{\alpha_n} \alpha_n g_n(w)$?

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\begin{aligned}
&\min_{\boldsymbol{w}} \; f(\boldsymbol{w}) \\
\text{s.t} \quad & g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N \\
& h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M
\end{aligned}$$

- Introduce Lagrange multipliers $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^N$, $\alpha_n \geq 0$, and $\boldsymbol{\beta} = \{\beta_m\}_{m=1}^M$, one for each constraint, and construct the following Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g_n(\boldsymbol{w}) + \sum_{m=1}^{N} \beta_n h_n(\boldsymbol{w})$$

- Consider $\mathcal{L}_P(\boldsymbol{w}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Note that
  - $\mathcal{L}_P(\boldsymbol{w}) = \infty$ if $\boldsymbol{w}$ violates any of the constraints (g's or h's)
  - $\mathcal{L}_P(\boldsymbol{w}) = f(\boldsymbol{w})$ if $\boldsymbol{w}$ satisfies all the constraints (g's and h's)

- Thus $\min_{\boldsymbol{w}} \mathcal{L}_P(\boldsymbol{w}) = \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ solves the same problem as the original problem and will have the same solution. For convex f, g, h, the order of min and max is interchangeable.

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\min_{\boldsymbol{w}} \; f(\boldsymbol{w})$$
$$\text{s.t} \quad g_n(\boldsymbol{w}) \leq 0, \quad n = 1, \ldots, N$$
$$h_m(\boldsymbol{w}) = 0, \quad m = 1, \ldots, M$$

- Introduce Lagrange multipliers $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^N$, $\alpha_n \geq 0$, and $\boldsymbol{\beta} = \{\beta_m\}_{m=1}^M$, one for each constraint, and construct the following Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g_n(\boldsymbol{w}) + \sum_{m=1}^{N} \beta_n h_n(\boldsymbol{w})$$

- Consider $\mathcal{L}_P(\boldsymbol{w}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Note that
  - $\mathcal{L}_P(\boldsymbol{w}) = \infty$ if $\boldsymbol{w}$ violates any of the constraints (g's or h's)
  - $\mathcal{L}_P(\boldsymbol{w}) = f(\boldsymbol{w})$ if $\boldsymbol{w}$ satisfies all the constraints (g's and h's)
- Thus $\min_{\boldsymbol{w}} \mathcal{L}_P(\boldsymbol{w}) = \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}\geq 0,\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ solves the same problem as the original problem and will have the same solution. For convex f, g, h, the order of min and max is interchangeable.
- Karush-Kuhn-Tucker (KKT) Conditions: At the optimal solution, $\alpha_n g_n(\boldsymbol{w}) = 0$ (note the $\max_\alpha$)

Solving Hard-Margin SVM

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

$$\boxed{\begin{aligned} &\min_{\boldsymbol{w},b} \quad f(\boldsymbol{w},b) = \frac{||\boldsymbol{w}||^2}{2} \\ &\text{subject to} \quad 1 - y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \leq 0, \qquad n = 1,\ldots,N \end{aligned}}$$

- A constrained optimization problem. Can solve using Lagrange's method

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

$$\boxed{\begin{array}{l} \min_{\boldsymbol{w},b} \quad f(\boldsymbol{w}, b) = \frac{||\boldsymbol{w}||^2}{2} \\ \text{subject to} \quad 1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \leq 0, \qquad n = 1, \ldots, N \end{array}}$$

- A constrained optimization problem. Can solve using Lagrange's method
- Introduce Lagrange Multipliers α_n $(n = \{1, \ldots, N\})$, one for each constraint, and solve the following Lagrangian:

$$\boxed{\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{||\boldsymbol{w}||^2}{2} + \sum_{n=1}^{N} \alpha_n \{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\}}$$

- Note: $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]$ is the vector of Lagrange multipliers

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

$$\boxed{\begin{aligned} &\min_{\boldsymbol{w},b} \quad f(\boldsymbol{w},b) = \frac{||\boldsymbol{w}||^2}{2} \\ &\text{subject to} \quad 1 - y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \leq 0, \qquad n = 1,\ldots,N \end{aligned}}$$

- A constrained optimization problem. Can solve using Lagrange's method
- Introduce Lagrange Multipliers α_n ($n = \{1,\ldots,N\}$), one for each constraint, and solve the following Lagrangian:

$$\boxed{\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}\geq 0} \mathcal{L}(\boldsymbol{w},b,\boldsymbol{\alpha}) = \frac{||\boldsymbol{w}||^2}{2} + \sum_{n=1}^{N} \alpha_n\{1 - y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b)\}}$$

- Note: $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_N]$ is the vector of Lagrange multipliers
- We will solve this Lagrangian by solving a dual problem (eliminate $\boldsymbol{w}$ and b and solve for the "dual variables" $\boldsymbol{\alpha}$)

Solving Hard-Margin SVM

- The original Lagrangian is

$$\boxed{\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}\geq 0} \mathcal{L}(\boldsymbol{w},b,\boldsymbol{\alpha}) = \frac{\boldsymbol{w}^\top \boldsymbol{w}}{2} + \sum_{n=1}^{N} \alpha_n\{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\}}$$

For convex function $\min_{w,b} \max_{\alpha} = \max_{\alpha} \min_{w}$

Interested students can further read on "strong duality".

Solving Hard-Margin SVM

- The original Lagrangian is

$$\boxed{\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{\boldsymbol{w}^\top \boldsymbol{w}}{2} + \sum_{n=1}^{N} \alpha_n \{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\}}$$

- Take (partial) derivatives of $\mathcal{L}$ w.r.t. $\boldsymbol{w}$, b and set them to zero

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boxed{\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n} \qquad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0$$

This means that $\vec{w}$ = weighted sum of I/p points

$\parallel^{r}$ to perceptron

Solving Hard-Margin SVM

- The original Lagrangian is

$$\boxed{\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}\geq 0} \ \mathcal{L}(\boldsymbol{w},b,\boldsymbol{\alpha}) = \frac{\boldsymbol{w}^\top \boldsymbol{w}}{2} + \sum_{n=1}^{N} \alpha_n \{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\}}$$

- Take (partial) derivatives of $\mathcal{L}$ w.r.t. $\boldsymbol{w}$, b and set them to zero

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boxed{\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n} \qquad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0$$

- Important: Note the form of the solution $\boldsymbol{w}$ - it is simply a weighted sum of all the training inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ (and α_n is like the "importance" of $\boldsymbol{x}_n$)
- Substituting $\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$ in Lagrangian and also using $\sum_{n=1}^{N} \alpha_n y_n = 0$

$$\boxed{\max_{\boldsymbol{\alpha}\geq 0} \ \mathcal{L}_D(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (\boldsymbol{x}_m^T \boldsymbol{x}_n) \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\boxed{\max_{\boldsymbol{\alpha} \geq 0} \quad \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

where $\mathbf{G}$ is an $N \times N$ matrix with $G_{mn} = y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$, and $\mathbf{1}$ is a vector of 1s

† If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin

Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\boxed{\max_{\boldsymbol{\alpha} \geq 0} \quad \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

  where $\mathbf{G}$ is an $N \times N$ matrix with $G_{mn} = y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$, and $\mathbf{1}$ is a vector of 1s

- **Good news:** This is maximizing a concave function (or minimizing a convex function - verify that the Hessian is $\mathbf{G}$, which is p.s.d.). Note that our original primal SVM objective was also convex

† If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin

Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\boxed{\max_{\boldsymbol{\alpha} \geq 0} \quad \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

where $\mathbf{G}$ is an $N \times N$ matrix with $G_{mn} = y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$, and $\mathbf{1}$ is a vector of 1s

- **Good news:** This is maximizing a concave function (or minimizing a convex function - verify that the Hessian is $\mathbf{G}$, which is p.s.d.). Note that our original primal SVM objective was also convex
- **Important:** Inputs $\boldsymbol{x}$'s only appear as inner products (helps to "kernelize")

† If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin

Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\boxed{\max_{\boldsymbol{\alpha} \geq 0} \quad \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

  where $\mathbf{G}$ is an $N \times N$ matrix with $G_{mn} = y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$, and $\mathbf{1}$ is a vector of 1s

- **Good news:** This is maximizing a concave function (or minimizing a convex function - verify that the Hessian is $\mathbf{G}$, which is p.s.d.). Note that our original primal SVM objective was also convex
- **Important:** Inputs $\boldsymbol{x}$'s only appear as inner products (helps to "kernelize")
- Can solve[†] the above objective function for $\boldsymbol{\alpha}$ using various methods, e.g.,

[†] If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin

Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\boxed{\max_{\alpha \geq 0} \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

  where $\mathbf{G}$ is an $N \times N$ matrix with $G_{mn} = y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$, and $\mathbf{1}$ is a vector of 1s

- **Good news:** This is maximizing a concave function (or minimizing a convex function - verify that the Hessian is $\mathbf{G}$, which is p.s.d.). Note that our original primal SVM objective was also convex

- **Important:** Inputs $\boldsymbol{x}$'s only appear as inner products (helps to "kernelize")

- Can solve† the above objective function for $\boldsymbol{\alpha}$ using various methods, e.g.,
  - Treating the objective as a Quadratic Program (QP) and running some off-the-shelf QP solver such as `quadprog` (MATLAB), CVXOPT, CPLEX, etc.

† If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin

Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\boxed{\max_{\boldsymbol{\alpha} \geq 0} \ \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0}$$

  where $\mathbf{G}$ is an $N \times N$ matrix with $G_{mn} = y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$, and $\mathbf{1}$ is a vector of 1s
- **Good news:** This is maximizing a concave function (or minimizing a convex function - verify that the Hessian is $\mathbf{G}$, which is p.s.d.). Note that our original primal SVM objective was also convex
- **Important:** Inputs $\boldsymbol{x}$'s only appear as inner products (helps to "kernelize")
- Can solve† the above objective function for $\boldsymbol{\alpha}$ using various methods, e.g.,
  - Treating the objective as a Quadratic Program (QP) and running some off-the-shelf QP solver such as `quadprog` (MATLAB), CVXOPT, CPLEX, etc.
  - Using (projected) gradient methods (projection needed because the α's are constrained). Gradient methods will usually be much faster than QP methods.

† If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin

Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T \boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T \boldsymbol{x}_n\right)$$

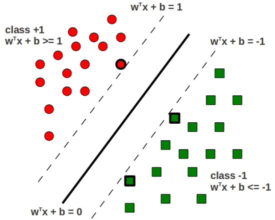
Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T \boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T \boldsymbol{x}_n\right)$$

- **A nice property:** Most α_n's in the solution will be zero (sparse solution)



  - Reason: Karush-Kuhn-Tucker (KKT) conditions

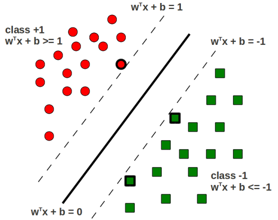
Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T\boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T\boldsymbol{x}_n\right)$$

- **A nice property:** Most α_n's in the solution will be zero (sparse solution)



- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal α_n's

$$\alpha_n\{1 - y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b)\} = 0$$

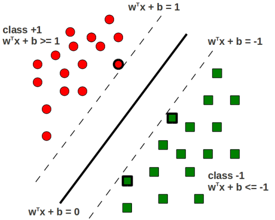
Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T\boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T\boldsymbol{x}_n\right)$$

- **A nice property:** Most α_n's in the solution will be zero (sparse solution)



- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal α_n's

$$\alpha_n\{1 - y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b)\} = 0$$

- α_n is non-zero only if $\boldsymbol{x}_n$

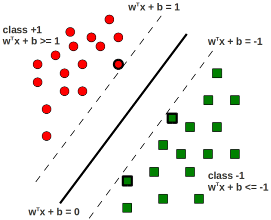
Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T \boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T \boldsymbol{x}_n\right)$$

- **A nice property:** Most α_n's in the solution will be zero (sparse solution)



- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal α_n's

$$\alpha_n\{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\} = 0$$

- α_n is non-zero only if $\boldsymbol{x}_n$ lies on one of the two margin boundaries, i.e., for which $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) = 1$

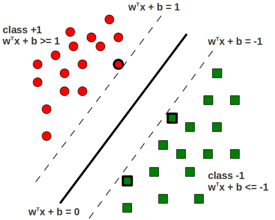
Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T \boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T \boldsymbol{x}_n\right)$$

- **A nice property:** Most α_n's in the solution will be zero (sparse solution)



- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal α_n's

$$\alpha_n\{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\} = 0$$

- α_n is non-zero only if $\boldsymbol{x}_n$ lies on one of the two margin boundaries, i.e., for which $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) = 1$
- These examples are called support vectors

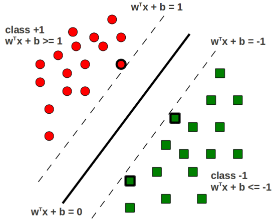
Hard-Margin SVM: The Solution

- Once we have the α_n's, $\boldsymbol{w}$ and b can be computed as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$$

$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \boldsymbol{w}^T \boldsymbol{x}_n + \max_{n:y_n=-1} \boldsymbol{w}^T \boldsymbol{x}_n\right)$$

- **A nice property:** Most α_n's in the solution will be zero (sparse solution)



- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal α_n's

$$\alpha_n\{1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b)\} = 0$$

- α_n is non-zero only if $\boldsymbol{x}_n$ lies on one of the two margin boundaries, i.e., for which $y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) = 1$
- These examples are called support vectors
- Recall the support vectors "support" the margin boundaries