

Learning Maximum-Margin Hyperplanes: Support Vector Machines



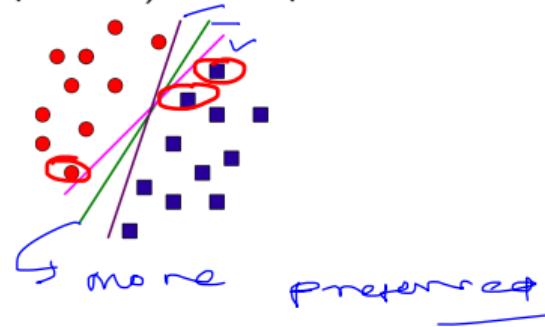
Machine Learning (CS771A)

Aug 24, 2016

Perceptron and (Lack of) Margins

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

2 - class

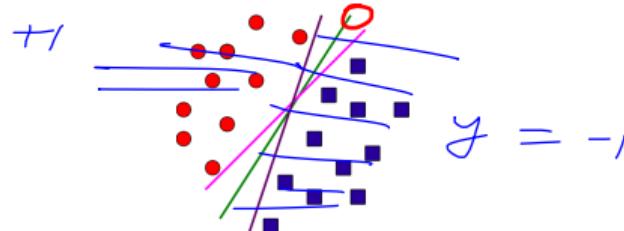


green — max degree of

sep

Perceptron and (Lack of) Margins

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

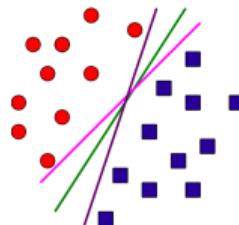


o Training
Error

- 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

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

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

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^2 a_i b_i = 2 + 6 = 8$$

$$= \frac{\|\vec{a}\| \|\vec{b}\| \cos \theta}{\sqrt{2}}$$

dot products and margin

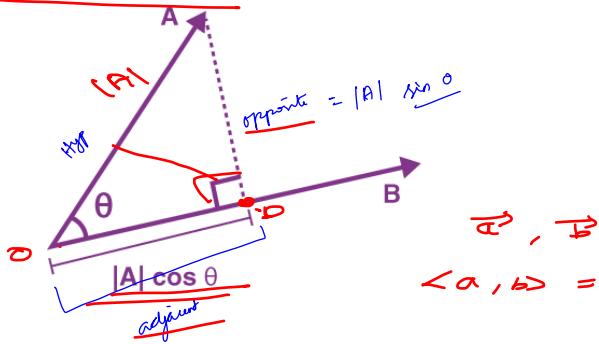
DOT PRODUCT OF VECTORS

BYJU'S
The Learning App

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

$$= \frac{OP}{OA}$$

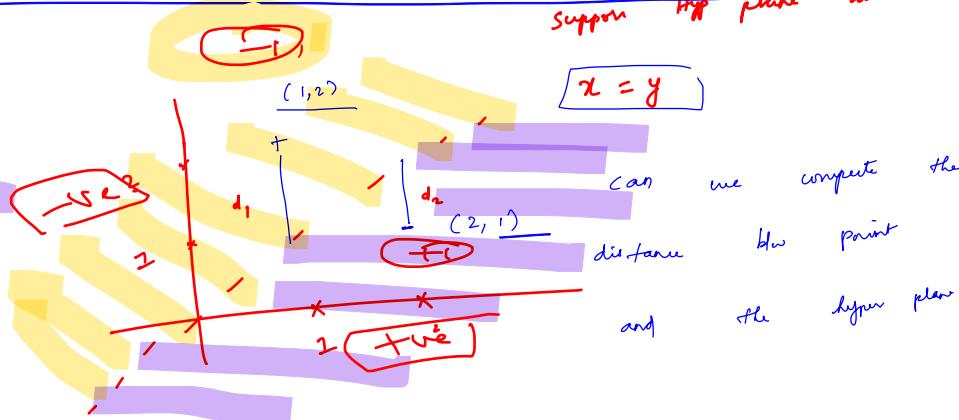
$$\cos \theta = \frac{OP}{OA}$$



Suppose the plane is

Semantics of

Signs ($CWT\alpha$)



Point - Line Distance

$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$

$$||w||$$

$$ax + by + c = 0$$

$$P(x_1, y_1)$$

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

$$\underline{a=1} \quad \text{and} \quad \underline{b=-1} \quad \underline{c=0}$$

d_1

d_2

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

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

dot product b/w line and point

denote \vec{w} as coefficients to the line

$$\vec{w} = \begin{bmatrix} x \\ y \\ z \\ a & b & c \end{bmatrix}$$

$$\boxed{P_1 = (1, 1, 1)} \\ \boxed{P_2 = (2, 1, 1)}$$

augment each point with 1 towards the

end to account for ' c '

$$x_1 = [1, 2, 1] \quad \text{and} \quad x_2 = [2, 1, 1]$$

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

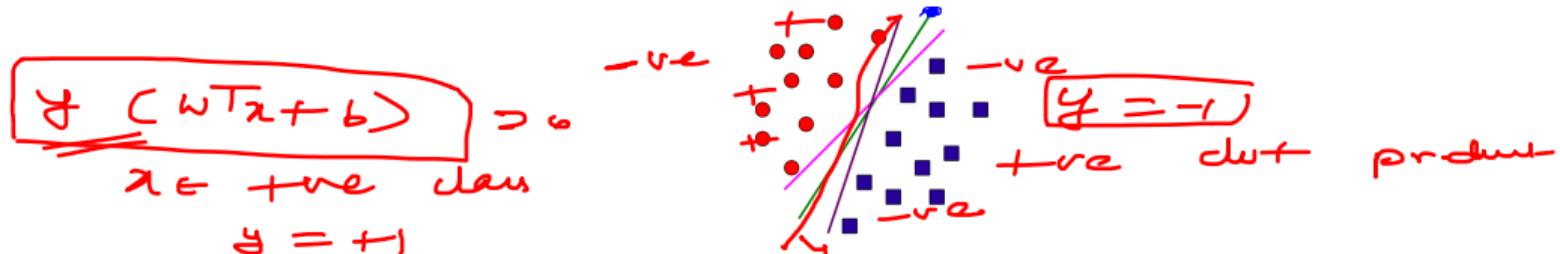
This is unnormalized signed distance.

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

Signs indicates the side of the margin

Perceptron and (Lack of) Margins

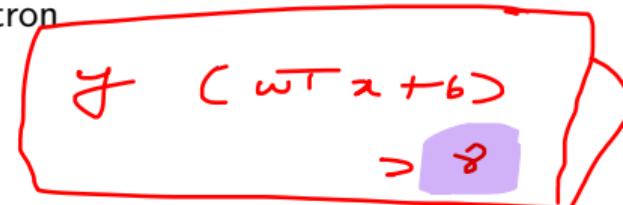
- Perceptron learns a hyperplane (of many possible) that separates the classes



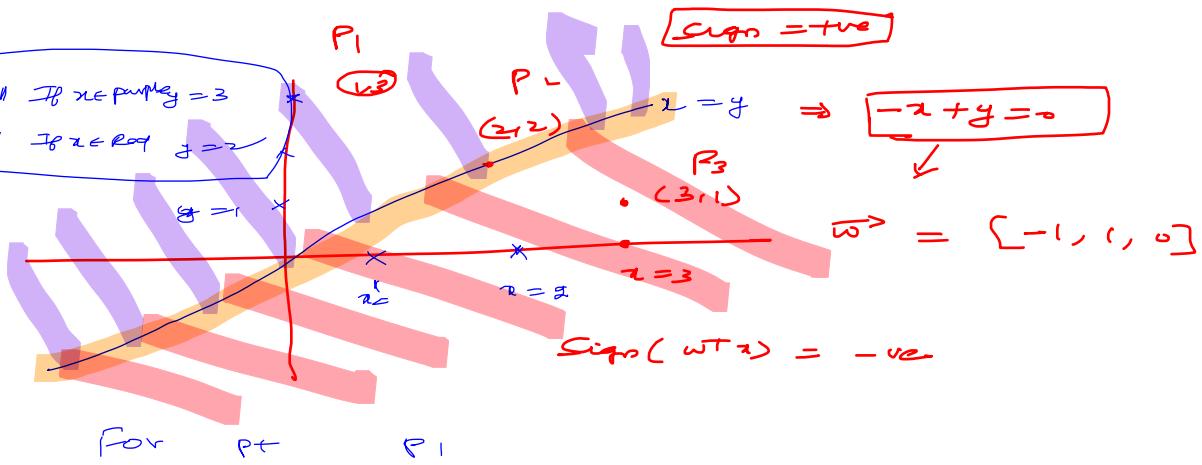
- Standard Perceptron doesn't guarantee any "margin" around the hyperplane
- Note: Possible to "artificially" introduce a margin in the Perceptron

- Simply change the Perceptron mistake condition to

$$y_n(w^T x_n + b) \leq \gamma$$



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



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

$$\begin{aligned} \text{For } P_2 \quad P_2 \\ w^T P_2 &= -2 + 2 = 0 \end{aligned} \quad \left\{ \begin{array}{l} w^T P_3 = -2 \\ \end{array} \right.$$

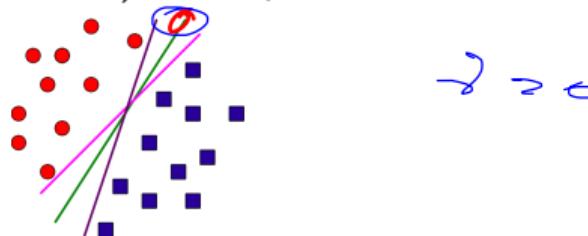
Solve for a \vec{w} such that

$$y = (w^T x + b) = ?$$

unnormalized distance

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
 - Simply change the Perceptron mistake condition to

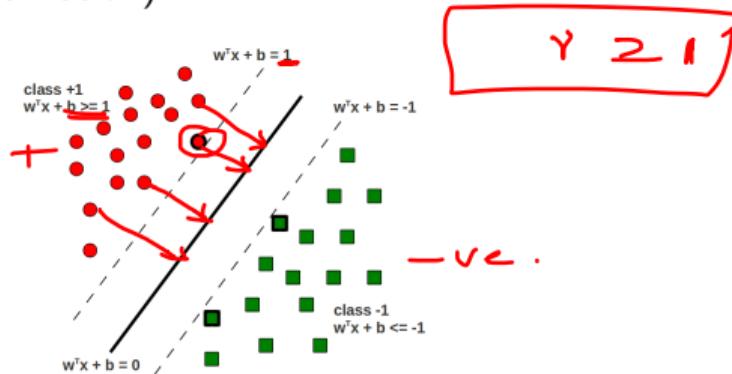
$$y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq \gamma$$

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

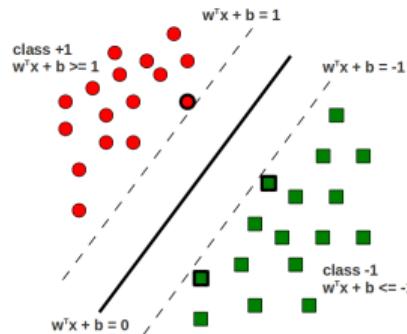
Support Vector Machine (SVM)

- 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)

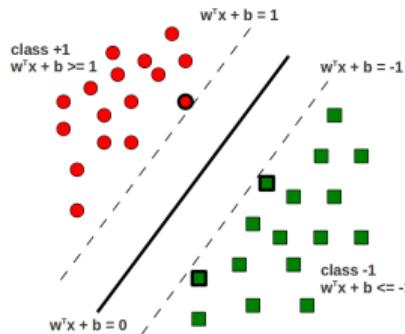
- Learns a hyperplane such that the positive and negative class training examples are **as far away as possible** from it (ensures good generalization)



- 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)

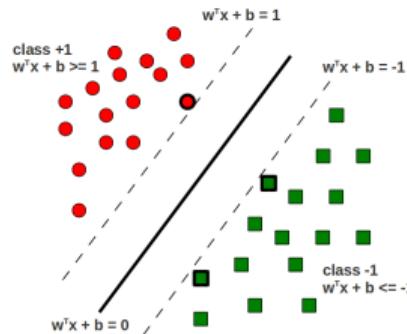
- Learns a hyperplane such that the positive and negative class training examples are **as far away as possible** from it (ensures good generalization)



- 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)

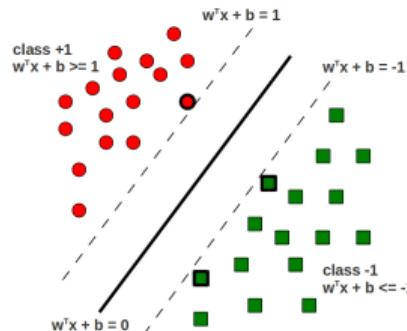
- Learns a hyperplane such that the positive and negative class training examples are **as far away as possible** from it (ensures good generalization)



- 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)

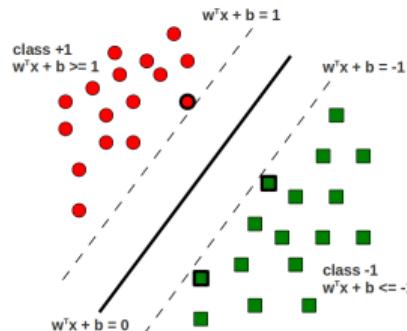
- Learns a hyperplane such that the positive and negative class training examples are **as far away as possible** from it (ensures good generalization)



- 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”).

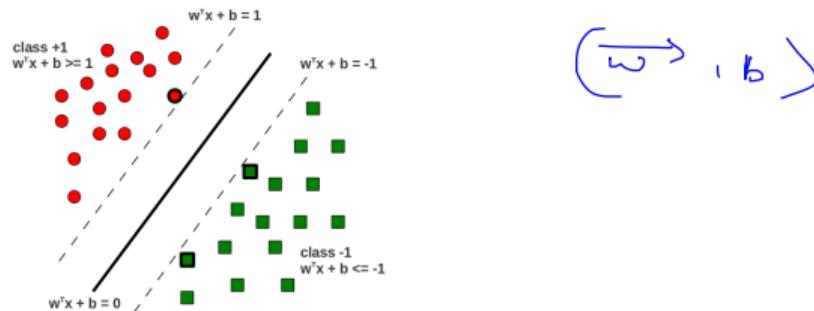
Support Vector Machine (SVM)

- 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

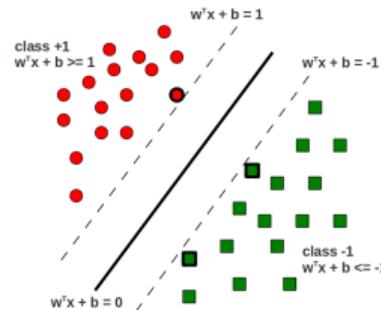


- Suppose there exists a hyperplane $w^T x + b = 0$ such that

- $w^T x_n + b \geq 1$ for $y_n = +1$
- $w^T x_n + b \leq -1$ for $y_n = -1$

$$\left\{ y_n (w^T x_n + b) \geq 1 \right.$$

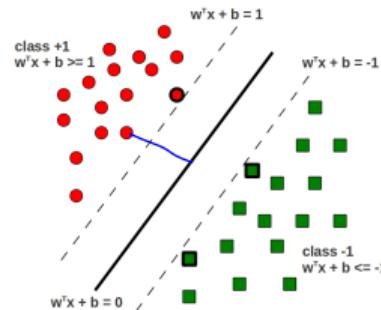
Learning a Maximum Margin Hyperplane



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

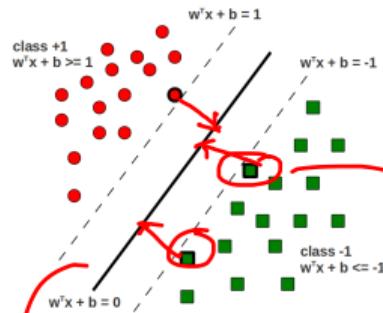


Learning a Maximum Margin Hyperplane



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

Learning a Maximum Margin Hyperplane



$$|w^T x + b| = \text{margin}$$

distance

- Suppose there exists a hyperplane $w^T x + b = 0$ such that

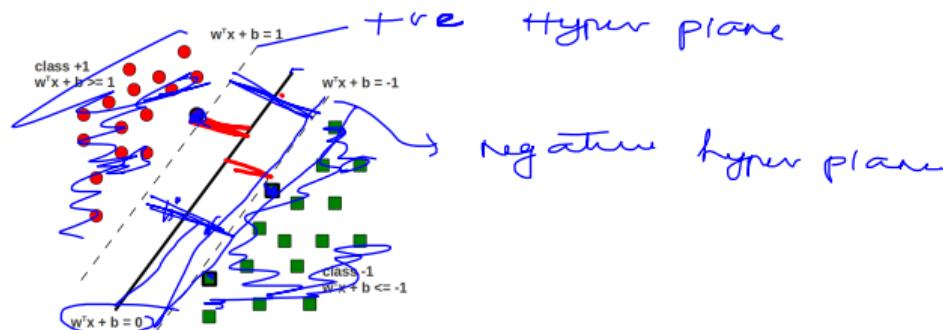
- $w^T x_n + b \geq 1$ for $y_n = +1$
- $w^T x_n + b \leq -1$ for $y_n = -1$
- Equivalently, $y_n(w^T x_n + b) \geq 1 \quad \forall n$ **(the margin condition)**
- Also note that $\min_{1 \leq n \leq N} |w^T x_n + b| = 1$
- Thus margin on each side: $\gamma = \min_{1 \leq n \leq N} \frac{|w^T x_n + b|}{\|w\|}$

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

st $y_n \in w^T x_n + b$

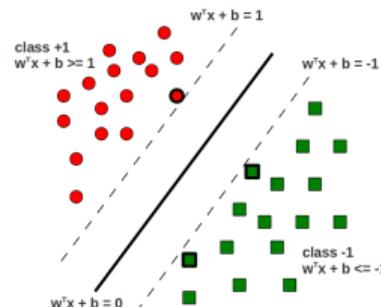
≥ 1

Learning a Maximum Margin Hyperplane



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

Learning a Maximum Margin Hyperplane



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

Large Margin = Good Generalization

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

Large Margin = Good Generalization

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

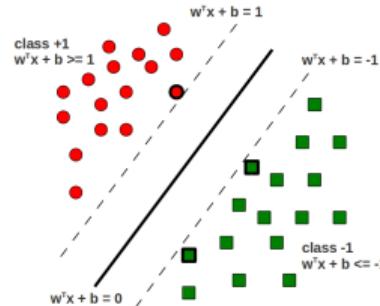
- Large margins intuitively mean good generalization
- We saw that margin $\gamma \propto \frac{1}{\|\mathbf{w}\|}$
- Large margin \Rightarrow small $\|\mathbf{w}\|$, i.e., small ℓ_2 norm of \mathbf{w}
- Small $\|\mathbf{w}\| \Rightarrow$ regularized/simple solutions (w_i 's don't become too large)
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

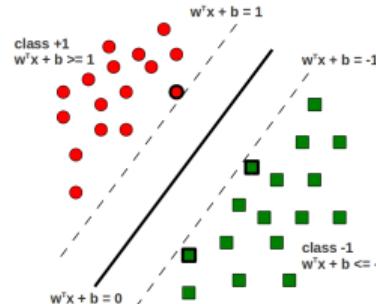
Hard-Margin SVM

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



Hard-Margin SVM

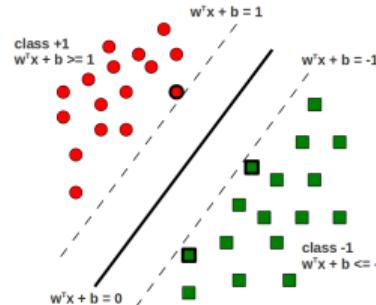
- Every training example has to fulfil the margin condition $y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$



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

Hard-Margin SVM

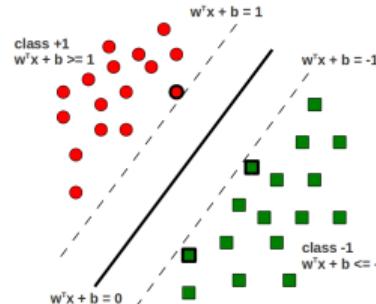
- Every training example has to fulfil the margin condition $y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$



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

Hard-Margin SVM

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



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

- Equivalent to minimizing $\|\mathbf{w}\|^2$ or $\frac{\|\mathbf{w}\|^2}{2}$

- The objective for hard-margin SVM

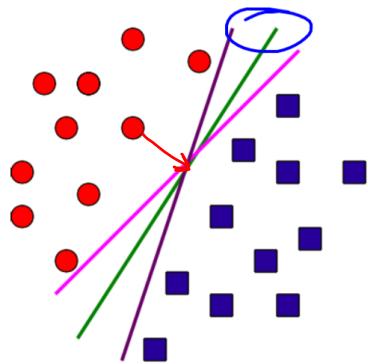
$$\min_{\mathbf{w}, b} f(\mathbf{w}, b) = \frac{\|\mathbf{w}\|^2}{2} \quad \xrightarrow{\text{quadratic loss function}}$$

subject to $y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N$

opt probm to be solved

function

linear constraint



Point - Line Distance

$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}} \quad ||w||$$

$$ax + by + c = 0$$

$$P(x_1, y_1)$$

$$ax + by + c = 0$$

$$\vec{w} = [a \ b \ c]$$

$$\vec{P} = [x_1 \ y_1 \ 1]$$

① Perceptron returns any arbitrary hyper plane

② we need to prefer green hyp plane.

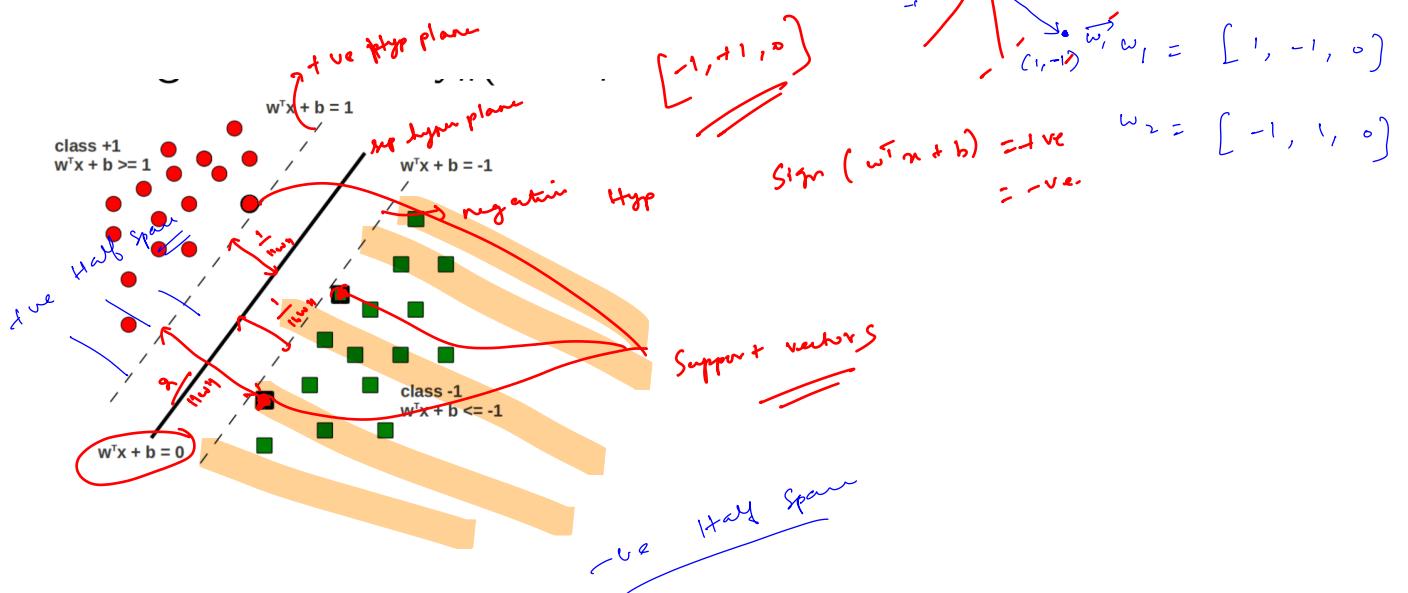
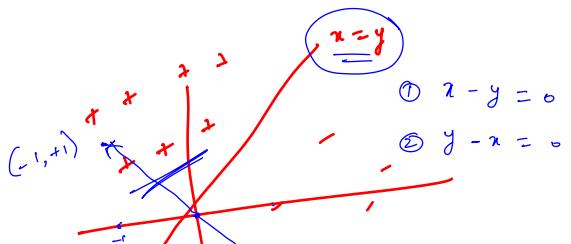
③ → need to compute the marg'n

margin (point x and hyp w) = $\frac{w^T x + b}{\|w\|}$

So the probm now is to find w so that

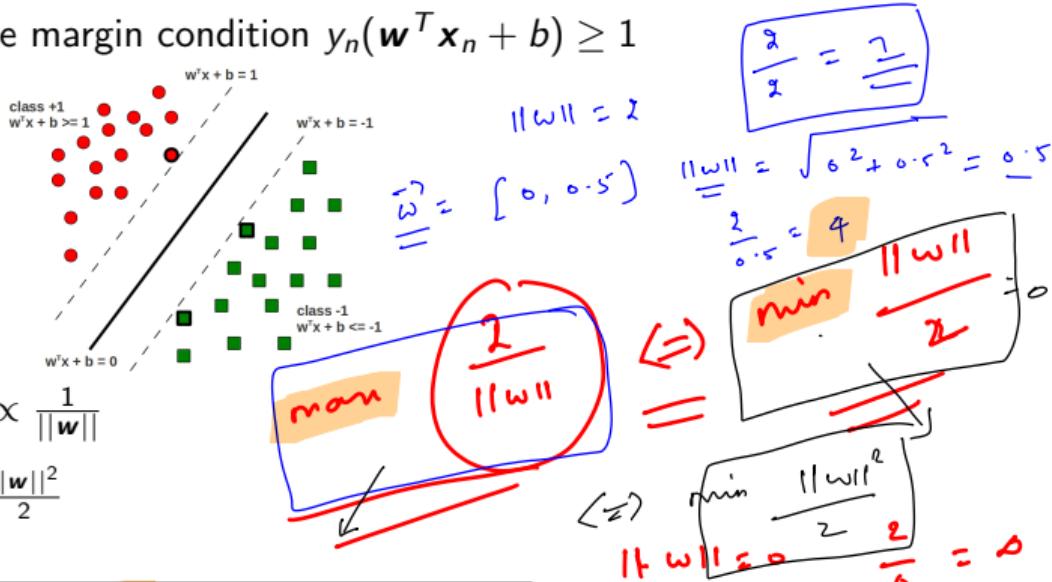
$$w^T x_i + b \geq \begin{cases} 1 & y_i = +1 \\ -1 & y_i = -1 \end{cases} \quad \Rightarrow \quad y_i (w^T x_i + b) \geq 1$$

margin = $\frac{w^T x + b}{\|w\|}$ ↪ optimization



Hard-Margin SVM

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



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

- Equivalent to minimizing $\|\mathbf{w}\|^2$ or $\frac{\|\mathbf{w}\|^2}{2}$

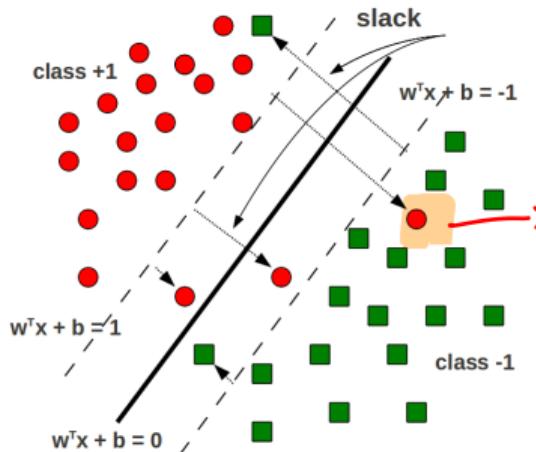
- The objective for hard-margin SVM

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

- 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



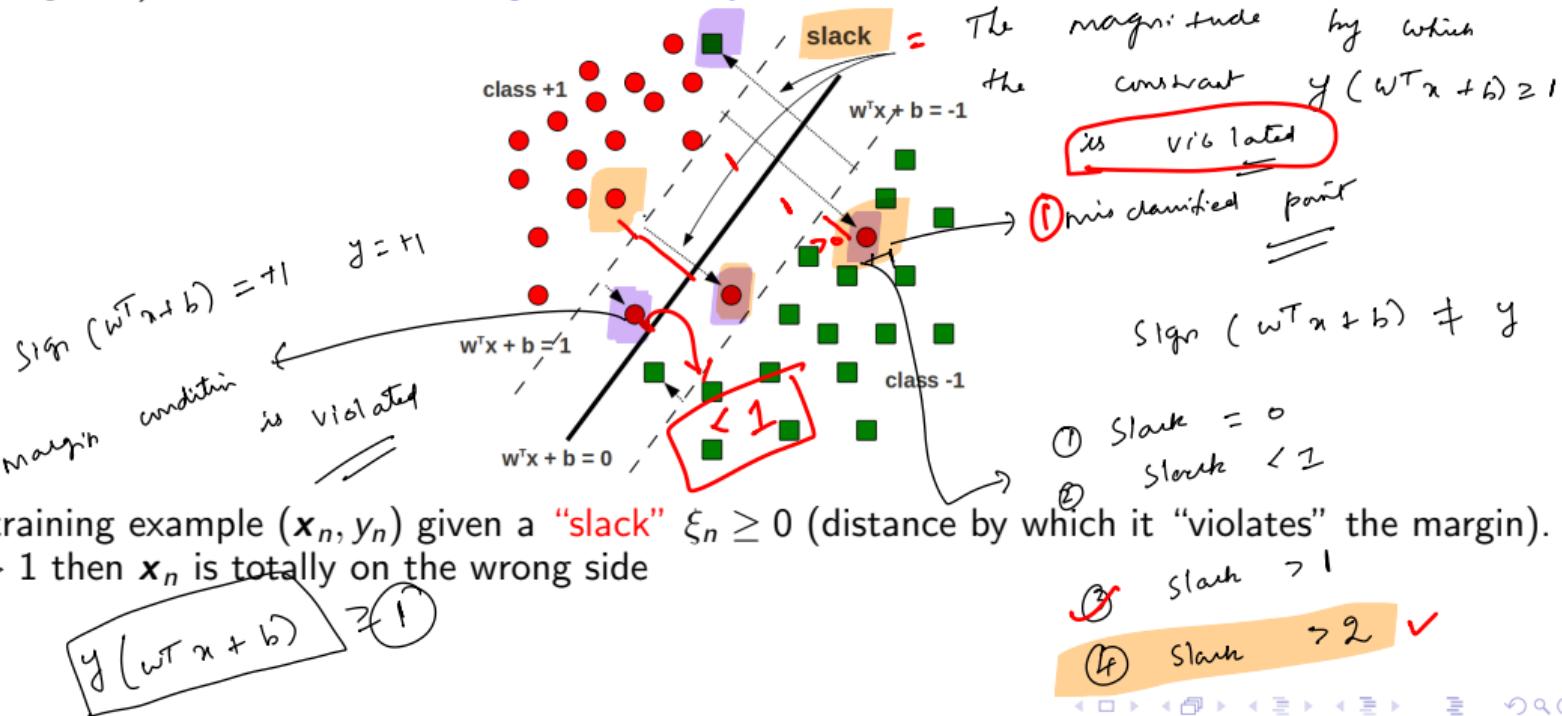
① has same / less Error
② max margin of Separation

suffer an Error!

Kernel Sum
→ hands on Session!

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



Slack

≤ 1

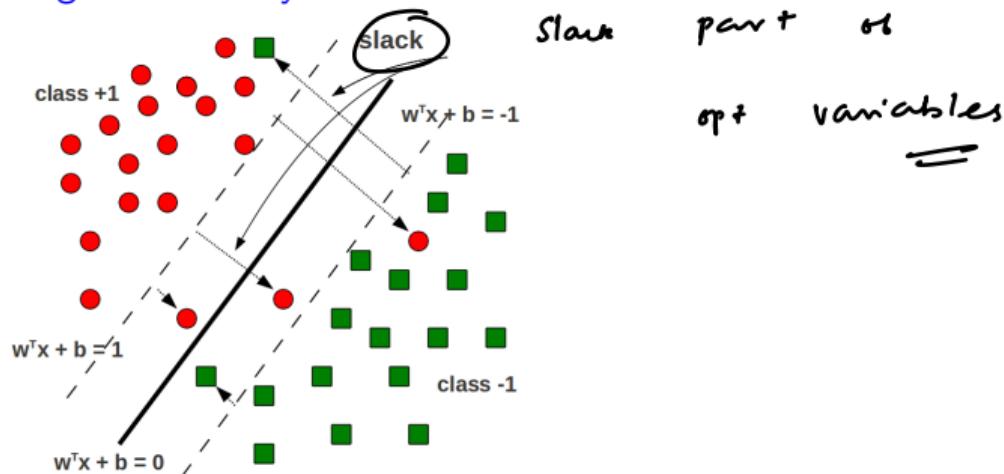
$\text{sign } (\omega^T x + b) = y \rightarrow$ True

slack ≥ 0

if $y_n (\omega^T x + b) \geq 1$ satisfied ?
 \hookrightarrow false!

Soft-Margin SVM (More Commonly Used)

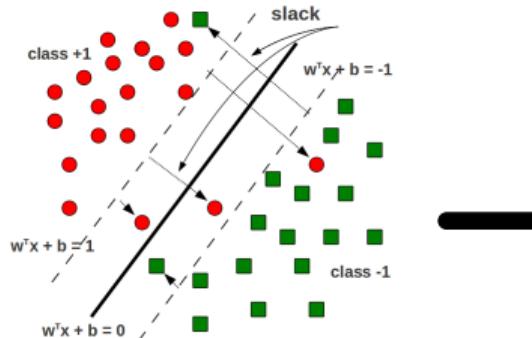
- Allow some training examples to fall **within** the margin region, or be even **misclassified** (i.e., fall on the wrong side). Preferable if training data is noisy



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

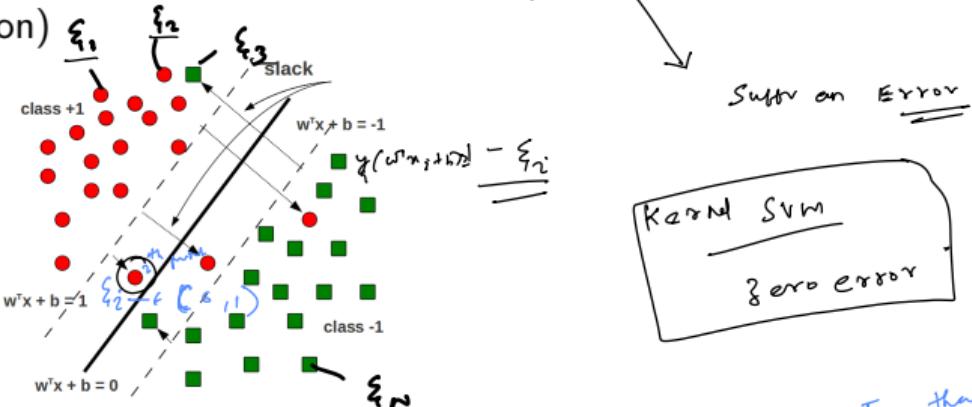
Soft-Margin SVM (More Commonly Used)

- 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

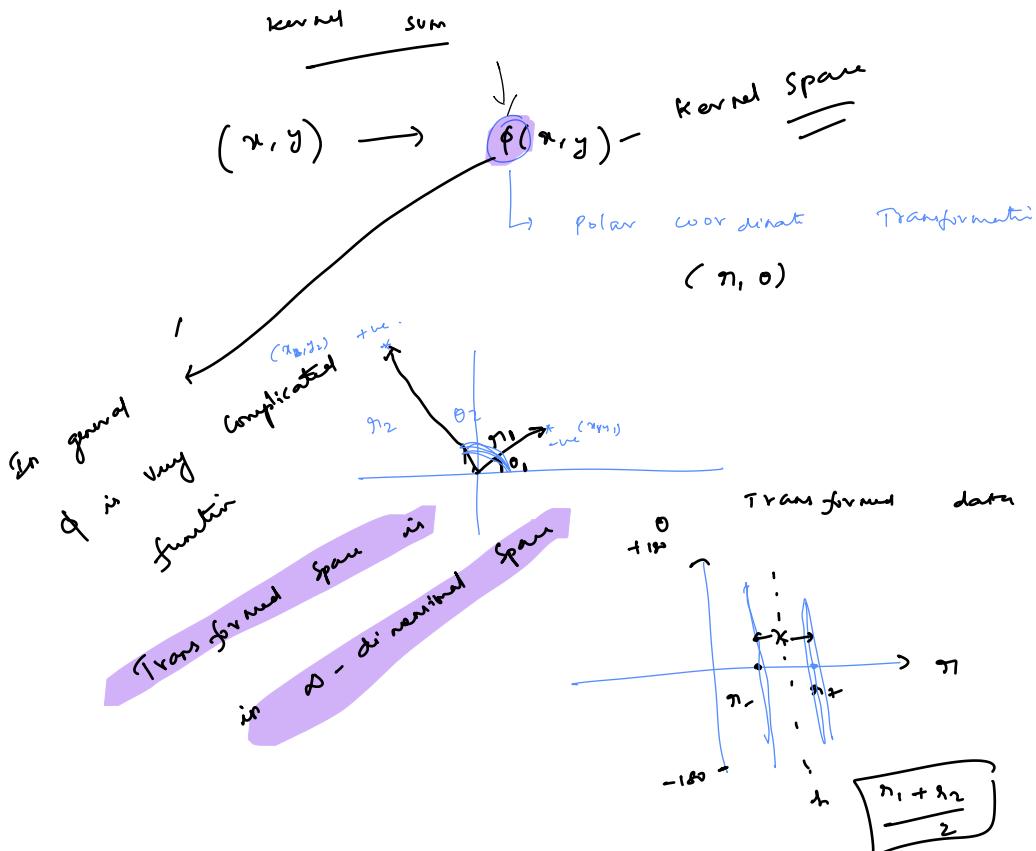
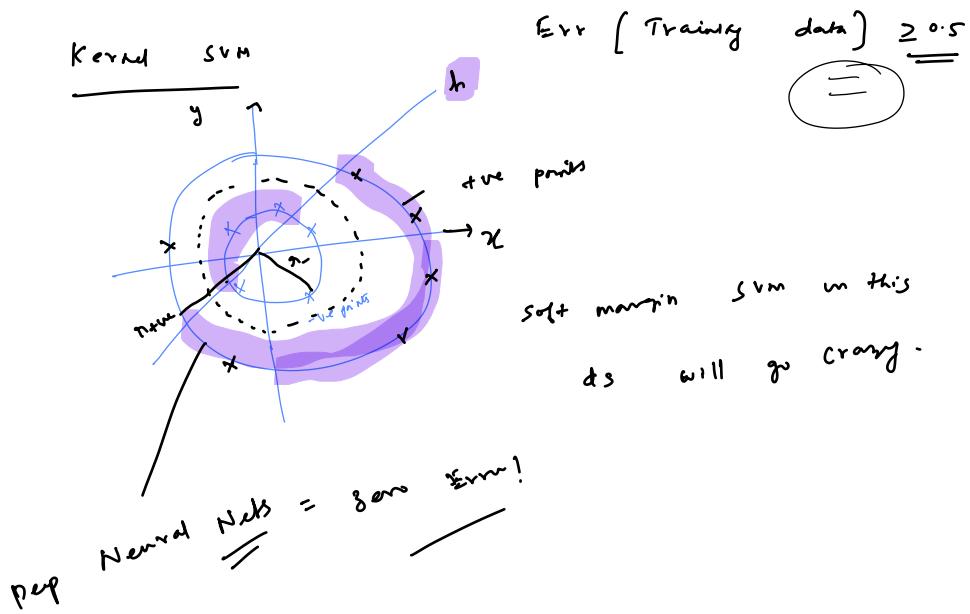
$$\min_{w, b, \xi} f(w, b, \xi) = \frac{\|w\|^2}{2} + C \sum_{n=1}^N \xi_n$$

subject to constraints $y_n(w^T x_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad n = 1, \dots, N$

Large margin

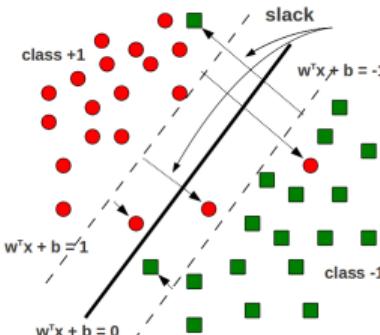
Hard margin SVM

for the points that
violating of
* margin condition
(Slack < 0)
* Label condition
(Slack $\neq 0$)



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)



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

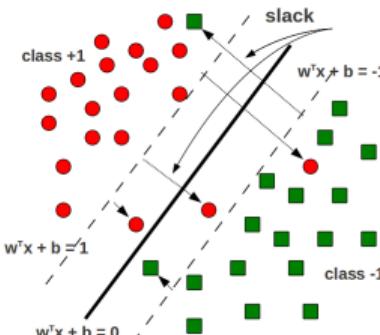
$$\min_{\mathbf{w}, b, \xi} f(\mathbf{w}, b, \xi) = \frac{\|\mathbf{w}\|^2}{2} + C \sum_{n=1}^N \xi_n$$

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

- 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)



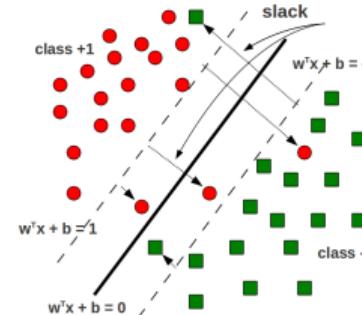
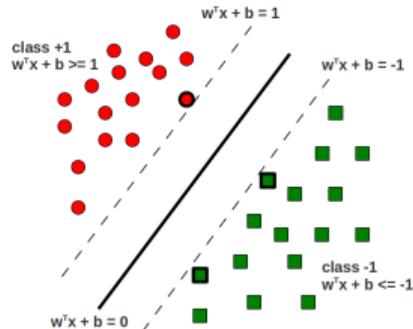
- The primal objective for soft-margin SVM can thus be written as

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

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

- 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 w and b)

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

subject to constraints $y_n(w^T x_n + b) \geq 1, \quad n = 1, \dots, N$

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

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

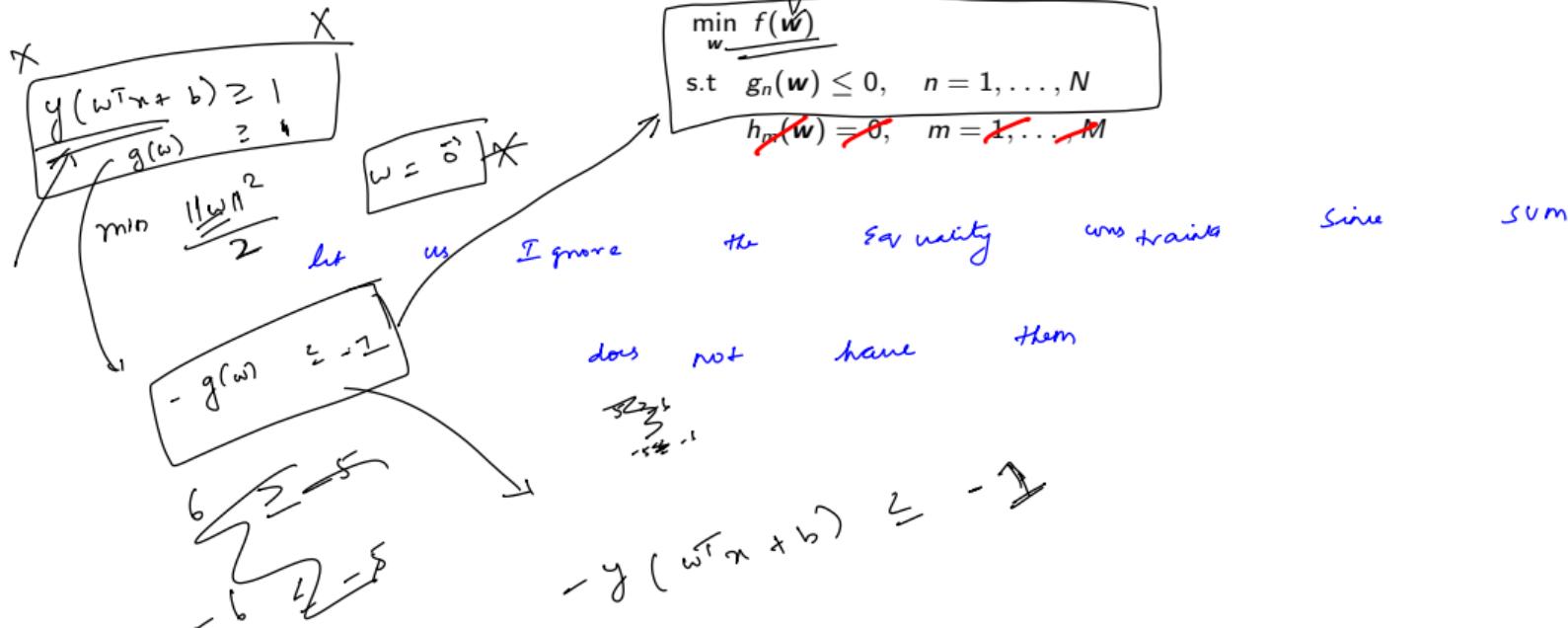
subject to $y_n(w^T x_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad n = 1, \dots, N$

- 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



Constrained Optimization via Lagrangian

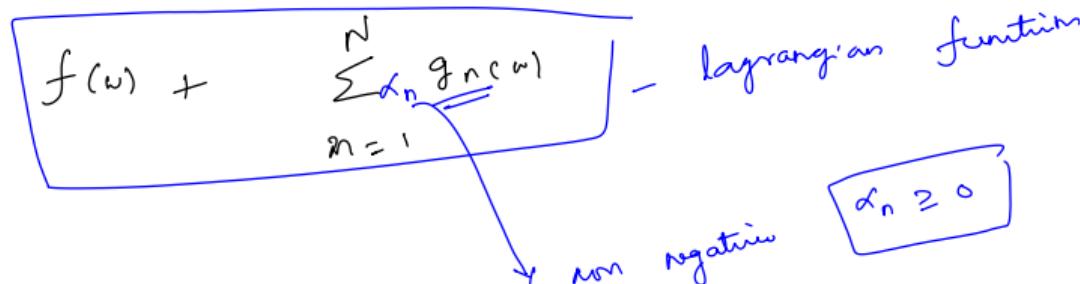
- Consider optimizing the following objective, subject to some constraints

$$\begin{array}{ll} \min_{\mathbf{w}, b} & f(\mathbf{w}) \\ \text{s.t.} & g_n(\mathbf{w}) \leq 0, \quad n = 1, \dots, N \\ & h_m(\mathbf{w}) = 0 \quad m = 1, \dots, M \end{array}$$

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

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{n=1}^N \alpha_n g_n(\mathbf{w}) + \sum_{m=1}^M \beta_m h_m(\mathbf{w})$$

$$\mathcal{L}(\mathbf{w}, b, \{\alpha_n\}_{n=1}^N)$$



Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

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

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

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{n=1}^N \alpha_n g_n(\mathbf{w}) + \sum_{m=1}^M \beta_m h_m(\mathbf{w})$$

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

$$\max_{\alpha} \mathcal{L}(\mathbf{w}, \alpha)$$

$$\min_{w, b} \frac{\|w\|^2}{2}$$

$$f(w) = \frac{\|w\|^2}{2}$$

$$\text{st } y_i (w^T x_i + b) \geq 1 \quad \forall i \in \Omega \subseteq N$$

$$-y_i (w^T x_i + b) \leq -1$$

$$-y_i (w^T x_i + b) + 1 \leq 0$$

$$L(w, b, \{x_i\}_{i=1}^n) = \frac{\|w\|^2}{2} + \sum_{i=1}^n \alpha_i (-y_i (w^T x_i + b) + 1)$$

dual problem

$$\max_{\{\alpha_i\}_{i=1}^n} L(w, b, \alpha)$$

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

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

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

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = \underbrace{f(\mathbf{w})}_{\leftarrow} + \sum_{n=1}^N \alpha_n \underbrace{g_n(\mathbf{w})}_{\leftarrow} + \sum_{m=1}^M \beta_m \underbrace{h_m(\mathbf{w})}_{\leftarrow}$$

w for which the constraint is violated

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

- $\mathcal{L}_P(\mathbf{w}) = \infty$ if \mathbf{w} violates any of the constraints (g 's or h 's)

unless $g_n(\mathbf{w})$ is violated

$$\Rightarrow g_n(\mathbf{w}) > 0$$

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

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

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\min_{\mathbf{w}} f(\mathbf{w})$$

$$\text{s.t. } g_n(\mathbf{w}) \leq 0, \quad n = 1, \dots, N$$

$$h_m(\mathbf{w}) = 0, \quad m = 1, \dots, M$$

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

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{n=1}^N \alpha_n g_n(\mathbf{w}) + \sum_{m=1}^M \beta_m h_m(\mathbf{w})$$

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

- $\mathcal{L}_P(\mathbf{w}) = \infty$ if \mathbf{w} violates any of the constraints (g 's or h 's) *If $g_n(\mathbf{w})$ is satisfied*
- $\mathcal{L}_P(\mathbf{w}) = f(\mathbf{w})$ if \mathbf{w} satisfies all the constraints (g 's and h 's)

$$g_n(\mathbf{w}) \leq 0$$

What is α_n in max $\alpha_n g_n(\mathbf{w})$?
 α_n

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

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

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

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{n=1}^N \alpha_n g_n(\mathbf{w}) + \sum_{m=1}^M \beta_m h_m(\mathbf{w})$$

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

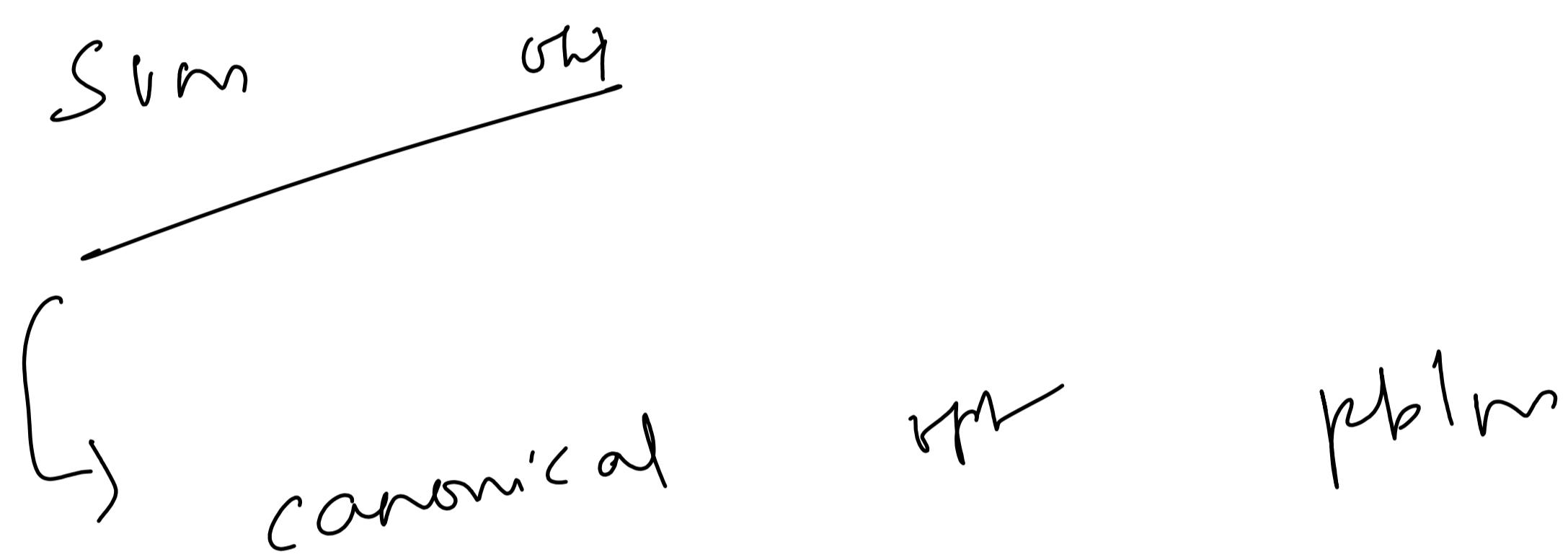
- $\mathcal{L}_P(\mathbf{w}) = \infty$ if \mathbf{w} violates any of the constraints (g 's or h 's)
 - $\mathcal{L}_P(\mathbf{w}) = f(\mathbf{w})$ if \mathbf{w} satisfies all the constraints (g 's and h 's)
- Thus $\min_{\mathbf{w}} \mathcal{L}_P(\mathbf{w}) = \min_{\mathbf{w}} \max_{\alpha \geq 0, \beta} \mathcal{L}(\mathbf{w}, \alpha, \beta)$ 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.

svm primal problem
is convex
why is quadratic
one for each constraint,
constraint are
linear equality const

$$\max_{\omega} \frac{1}{\|\omega\|}$$

$$\min_{\omega} \frac{\|\omega\|^2}{2}$$

21



$$\min_{\omega, b} f(\omega)$$

ω, b

$\sum_{i=1}^n g_i(\omega) \leq 0 \quad \forall n \in \text{Training data}$

$\boxed{\alpha_i = 0} \quad \text{if } g_i(\omega) > 0$

$\boxed{\alpha_i = 1} \quad \text{if } g_i(\omega) < 0$

① Lagrangian

$$\max_{\lambda} L(\omega, \lambda)$$

$$f(\omega) + \sum_{i=1}^n \alpha_i g_i(\omega)$$

constraint violated
for i^{th} point

$$\alpha_i \geq 0$$

$$g_i(\omega) > 0$$

$$\alpha_i > 0 \neq 1$$

dual

$$\max_{\lambda} L(\lambda, \omega) = +\infty \quad \text{there are } \geq 1$$

constraint violated

$\min_{\omega} \max_{\lambda} L(\lambda, \omega)$

Constrained Optimization via Lagrangian

- Consider optimizing the following objective, subject to some constraints

$$\min_{\mathbf{w}} f(\mathbf{w})$$

$$\text{s.t. } g_n(\mathbf{w}) \leq 0, \quad n = 1, \dots, N$$

$$h_m(\mathbf{w}) = 0, \quad m = 1, \dots, M$$

durability gap = 0

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

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{n=1}^N \alpha_n g_n(\mathbf{w}) + \sum_{m=1}^M \beta_m h_m(\mathbf{w})$$

$\mathcal{L}(\mathbf{w}, \cdot)$

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

- $\mathcal{L}_P(\mathbf{w}) = \infty$ if \mathbf{w} violates any of the constraints (g 's or h 's)
- $\mathcal{L}_P(\mathbf{w}) = f(\mathbf{w})$ if \mathbf{w} satisfies all the constraints (g 's and h 's)

min \mathbf{w} max α β $\mathcal{L}(\mathbf{w}, \alpha, \beta)$ \Leftarrow $\mathcal{L}_P(\mathbf{w})$ ≤ 0 $g_n(\mathbf{w}) \leq 0$

- Thus $\min_{\mathbf{w}} \mathcal{L}_P(\mathbf{w}) = \min_{\mathbf{w}} \max_{\alpha \geq 0, \beta} \mathcal{L}(\mathbf{w}, \alpha, \beta)$ 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, $\alpha_n g_n(\mathbf{w}) = 0$ (note the \max_{α})

$\forall n \quad w^*$ optimal

$$\alpha_n g_n(w) = 0$$

$g_n(w) > 0$ constraint
violation

$$0 = g_n(w) < 0 \quad \alpha_n = 0$$

$$0 = \underline{g_n(w)} = 0 \quad \alpha_n \text{ is any}$$

non zero
number.

Solving Hard-Margin SVM

Complementary Slackness condition

$$\alpha_n g_n(w) = 0$$

only at feasible / optimal

$$w \neq 0$$

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

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



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

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

Primal pbm

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

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

Primal pbm

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

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

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

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

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

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

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

Solving Hard-Margin SVM

- The original Lagrangian is

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

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

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

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

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

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \alpha_n y_n = 0$$

↓
This means that $\vec{\omega}$ = weighted sum
of all points

$\|\vec{\omega}\|$ to perception

Solving Hard-Margin SVM

- The original Lagrangian is

max w, b

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

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

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^N \alpha_n y_n x_n$$
$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \alpha_n y_n = 0$$

wt a
solution

for
 b'

- Important: Note the form of the solution w - it is simply a **weighted sum of all the training inputs** x_1, \dots, x_N (and α_n is like the "importance" of x_n)

- Substituting $w = \sum_{n=1}^N \alpha_n y_n x_n$ in Lagrangian and also using $\sum_{n=1}^N \alpha_n y_n = 0$

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

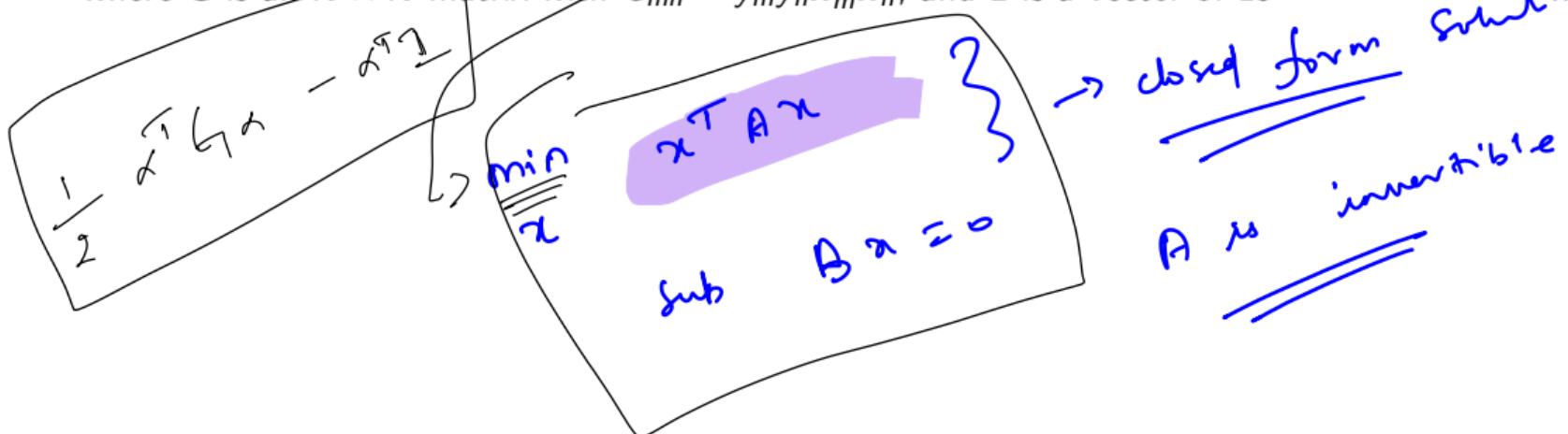
into a
constraint

Solving Hard-Margin SVM

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

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

where \mathbf{G} is an $N \times N$ matrix with $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$, 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

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

where \mathbf{G} is an $N \times N$ matrix with $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$, 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

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

where \mathbf{G} is an $N \times N$ matrix with $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$, 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 \mathbf{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

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

where \mathbf{G} is an $N \times N$ matrix with $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$, 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 \mathbf{x} 's only appear as **inner products** (helps to “kernelize”)
- Can solve[†] the above objective function for α 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

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

where \mathbf{G} is an $N \times N$ matrix with $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$, 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 \mathbf{x} 's only appear as **inner products** (helps to “kernelize”)
- Can solve[†] the above objective function for α 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

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

where \mathbf{G} is an $N \times N$ matrix with $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$, 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 \mathbf{x} 's only appear as **inner products** (helps to "kernelize")
- Can solve[†] the above objective function for α 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, \mathbf{w} and b can be computed as:

A handwritten derivation of the SVM solution. It starts with the formula for \mathbf{w} :

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

Below it is the formula for b :

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

Annotations include:

- A blue bracket groups the terms $\alpha_n y_n \mathbf{x}_n$ and $\mathbf{w}^T \mathbf{x}_n$.
- A handwritten note "comes from data" with arrows pointing to the summation in \mathbf{w} and the min/max terms in b .
- A handwritten note "Quadratic" with arrows pointing to the α_n term in \mathbf{w} and the $\mathbf{w}^T \mathbf{x}_n$ terms in b .

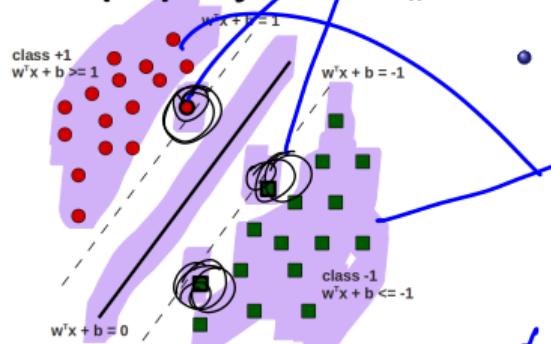
Hard-Margin SVM: The Solution

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

$$\mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n$$
$$b = -\frac{1}{2} (\min_{n:y_n=+1} \mathbf{w}^T \mathbf{x}_n + \max_{n:y_n=-1} \mathbf{w}^T \mathbf{x}_n)$$

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

in α -variables



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

compo' necessary Slackness condition

$$\alpha \approx 0$$
$$\nabla_w g_n(w) = 0$$

KKT

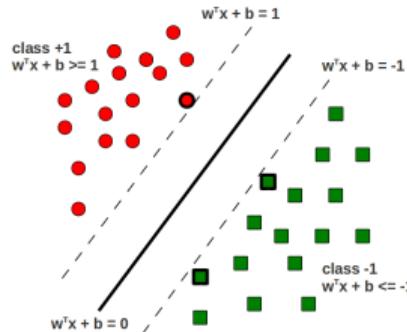
Hard-Margin SVM: The Solution

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

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

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

- 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

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

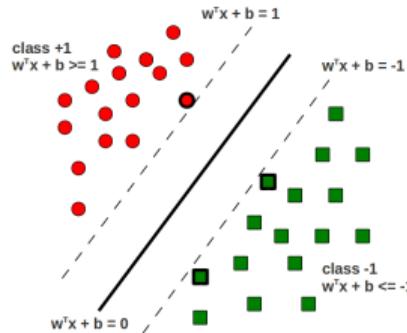
Hard-Margin SVM: The Solution

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

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

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

- 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
- $$\alpha_n \{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)\} = 0$$
- α_n is non-zero only if \mathbf{x}_n

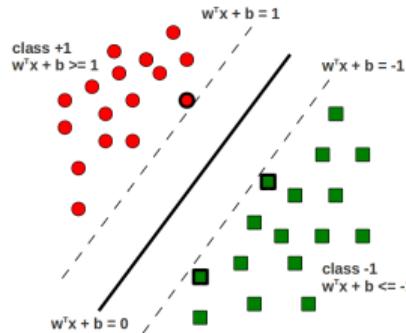
Hard-Margin SVM: The Solution

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

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

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

- 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
- $$\alpha_n \{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)\} = 0$$
- α_n is non-zero only if \mathbf{x}_n lies on one of the two margin boundaries, i.e., for which $y_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$

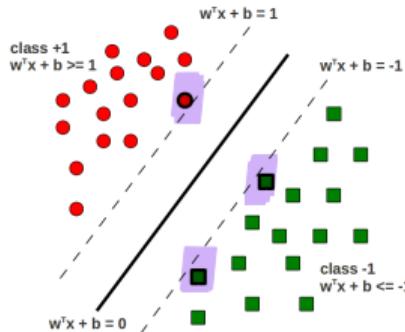
Hard-Margin SVM: The Solution

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

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

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

- 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

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

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

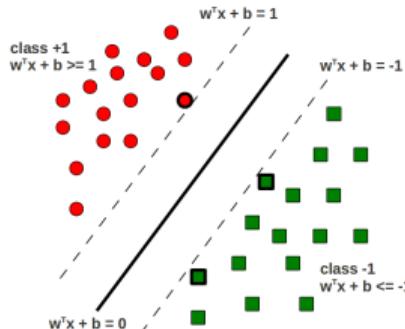
Hard-Margin SVM: The Solution

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

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

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

- 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

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

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

Kernels

Original data in XY plane

↪ not linearly separable.

ϕ : Transformation function

$\phi(\text{Training data}) = \text{Transformed data}$

apply SVM in the kernel space

Hope is that data is linearly separable in kernel space

ϕ : polar coordinate transformation

(r, θ) Transformation

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

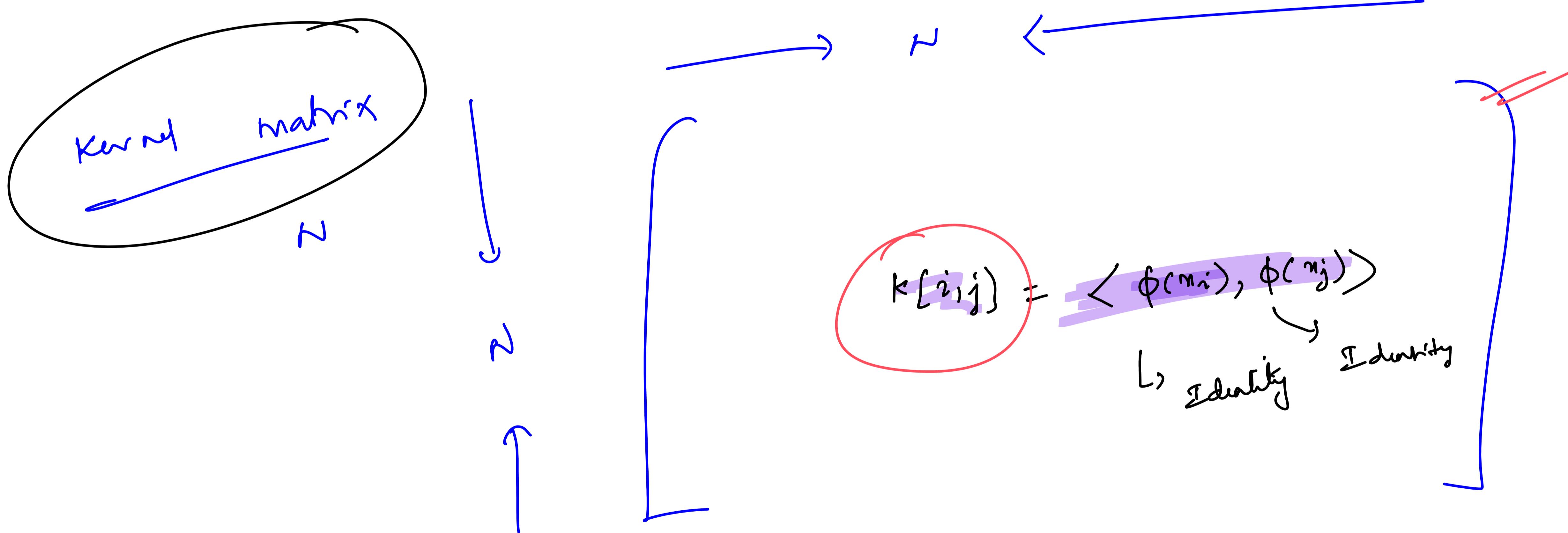
dot product kernel - linear kernel $\phi = \text{Identity} = \langle x_i, x_j \rangle$

$$\boxed{\phi = \text{Id}}$$

3 dim feature space $x_i = [x_{i1} \ x_{i2} \ x_{i3}]$

$$\langle x_i \cdot x_j \rangle = \sum_{k=1}^3 w_{ik} x_{jk} - \text{linear kernel}$$

$\rightarrow N \leftarrow$



poly nomial kernel

1-d coordinate

$n \in \mathbb{R}$

$x = 5 \quad y = 7$

$$k(x_i, y) = \|x_i - y\|_2^2$$

$$\Leftrightarrow k(5, 7) = (7 - 5)^2 = 4$$

this is not definition

for polynomial kernel

(x_i, y)

$$k_{ij} = \|x_i - x_j\|_2^2$$

\hookrightarrow is a transformation also simple?

Poly

$$\underline{x + r^2}$$

$$x = [1, 2] \quad y = [2, 3]$$

$$k(x, y) = \underline{\|x - y\|^2} = \left(\sqrt{(1-2)^2 + (2-3)^2} \right)^2 = 2$$

Solving kernel value $\underline{g_m}(x, y)$ is easy -

$\underline{\phi}$ transfer matrix

$$\underline{\|x - y\|_2^2} = \left\| \begin{bmatrix} x_1, x_2 \\ y_1, y_2 \end{bmatrix} \right\|^2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

$$= x_1^2 + y_1^2 - 2x_1y_1 + x_2^2 + y_2^2 - 2x_2y_2$$

Is it possible to rep the kernel operation in the form

$$\underline{\langle \phi(x), \phi(y) \rangle}$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{pmatrix} \langle \phi(x), \phi(y) \rangle = \begin{pmatrix} y_1 \\ y_2 \\ \sqrt{2}y_1 \\ \sqrt{2}y_2 \end{pmatrix}$$

Polynomial Kernel

Linear Kernel

$d = 1$

$$k(x, y) = (\langle x, y \rangle)^d$$

$d = 100$

assume $x, y \in \mathbb{R}^n$ and $d = 2$ for simplicity

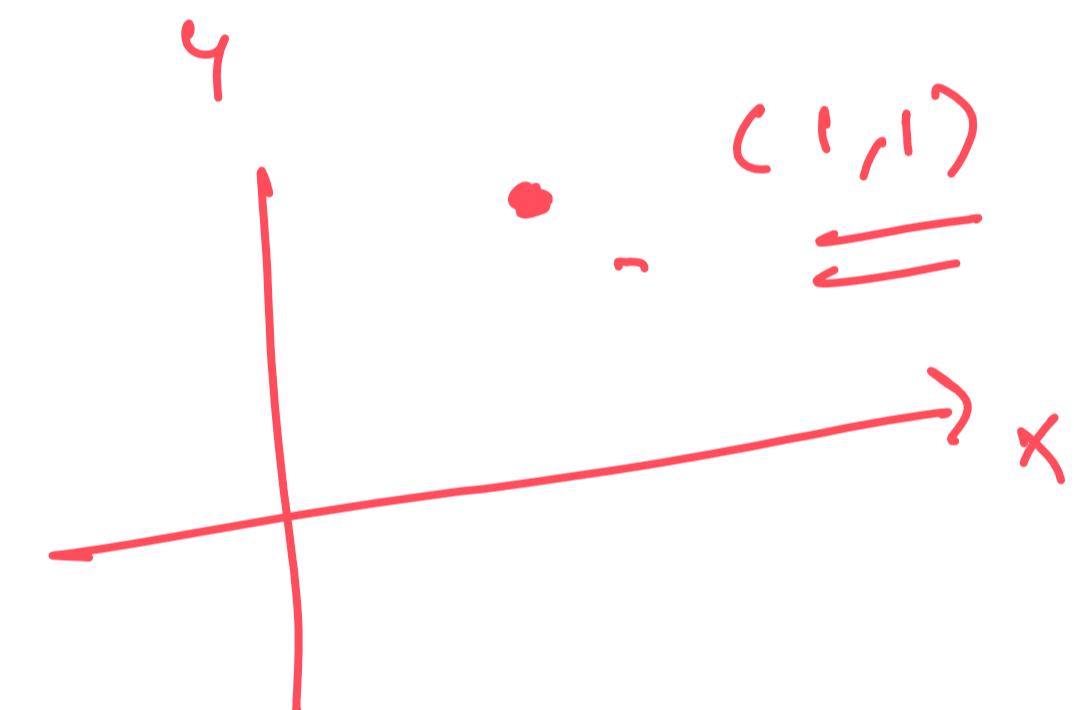
$$k(x, y) = (x_1 y_1 + x_2 y_2)^{100}$$

$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 x_2 y_2$$

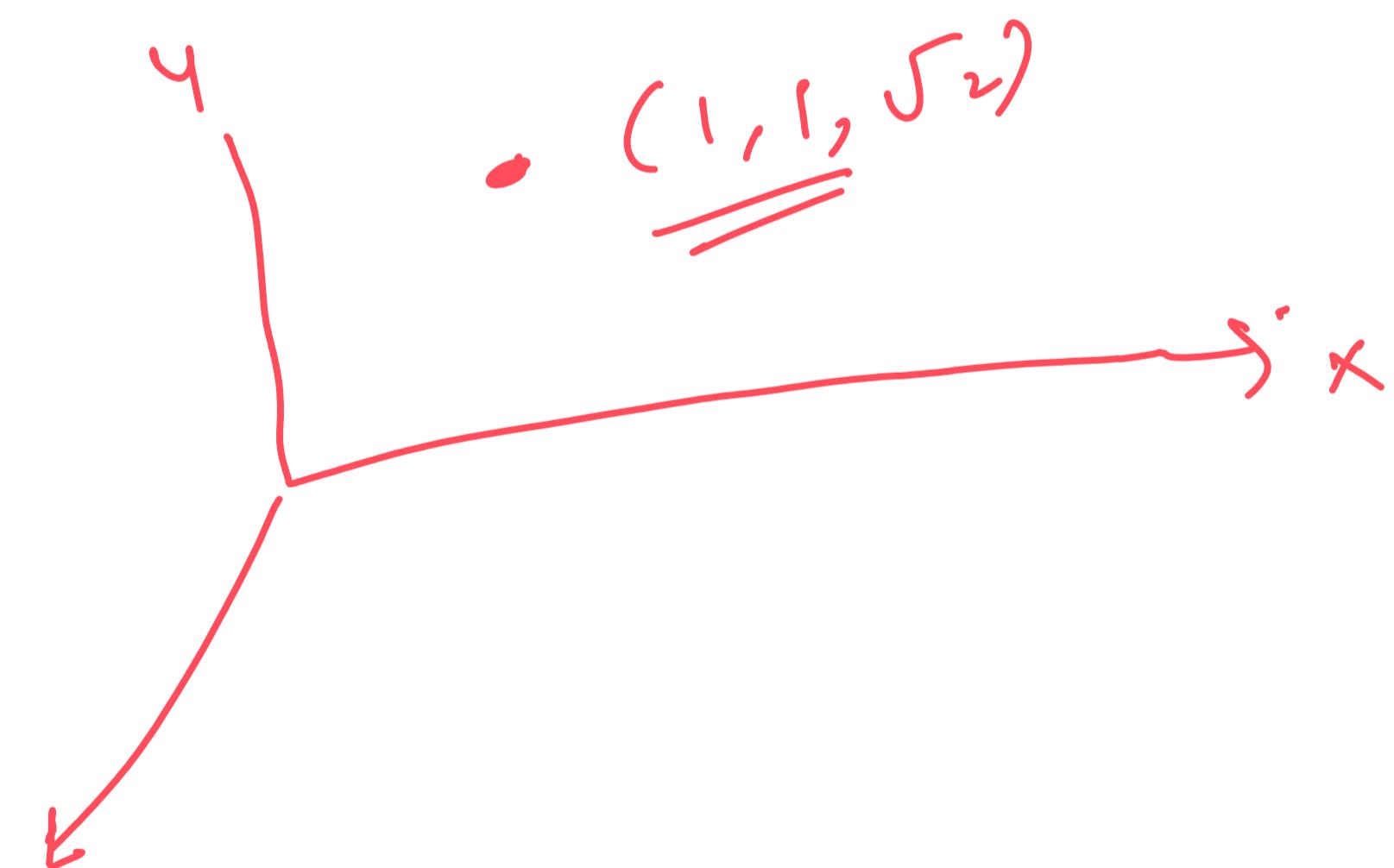
$$= \langle (x_1^2, x_2^2, \sqrt{2}x_1 x_2), (y_1^2, y_2^2, \sqrt{2}y_1 y_2) \rangle$$

$$\phi(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix}$$

so given pt in \mathbb{R}^2 space



$$\phi(x) = \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \end{pmatrix}$$



SVM solves in 3d space

$$k(x, y) = (\langle x, y \rangle)^d \Rightarrow \text{SVM solves in } d \text{ dims space}$$

RBF kernel

most powerful kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2}\right)$$

— Gaussian distribution

From Taylor Expansion of e^x around $x = 0$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \quad \text{2 terms}$$

$$e^{-\frac{1}{2} \|x-y\|^2} = e\left(-\frac{1}{2} (\|x\|^2 + \|y\|^2 - 2\langle x, y \rangle)\right)$$

$$= e^{-\frac{1}{2} \|x\|^2} \cancel{+} e^{-\frac{1}{2} \|y\|^2} \cancel{*} e^{-\langle x, y \rangle}$$

constant and can be ignored

$$= e^{-\langle x, y \rangle}$$

$\cancel{+}$ $\cancel{*}$ $\cancel{\cdot}$ $\cancel{=}$ $\cancel{\delta}$ - dimension