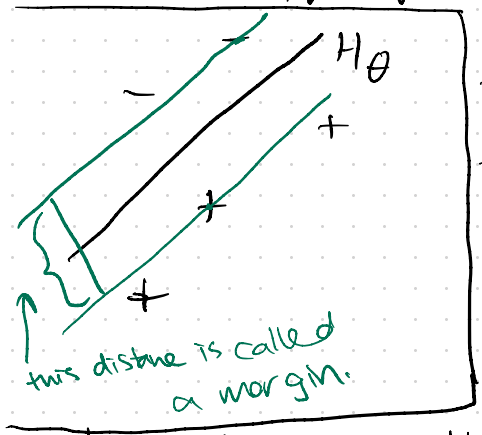
Recall from last lecture,

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^D \times \underbrace{\{1, -1, 0\}}_{\text{binary classification}}^N$$

If data is linearly separable, that is

$\exists$ Hyperplane $H = \{z \mid f_\theta(z) = 0\}$ with



$f_\theta(z) = \langle z, \theta' \rangle + \theta$ so that

$y_i = +1$ on one side,

and $y_i = -1$ on other side of H_θ.

Linearly separable data admits a solution to the Hard margin SVM:

$$\max_{\substack{\theta \in \mathbb{R}^{D+1} \\ \|\theta'\| = 1}} \left\{ r \geq 0 \;\middle|\; \operatorname{dist}(x_k, H_\theta) \geq r \right\}_{k=1,\ldots,n}$$

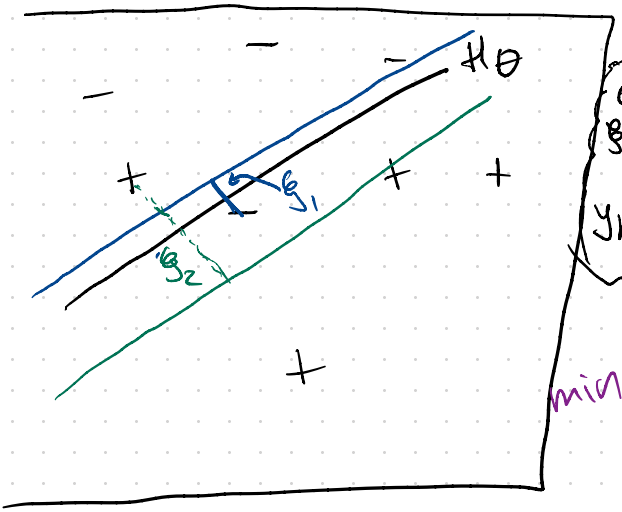
Dividing by r, equivalently, Hard margin SVM is

$$\min_{\theta \in \mathbb{R}^{D+1}} \left\{ \|\theta'\|^2 \;\middle|\; y_k(\langle \theta', x_k \rangle + \theta_0) \geq 1 \right\}_{k=1,\ldots,n}$$

Soft margin SVM

still linear structure, but adds "optimal" band of error.

$C > 0$ is a fixed regularization variable.



$$\min_{\theta \in \mathbb{R}^{n+1},\ \xi_1,\dots,\xi_k} \|\theta'\|^2 + C\sum_{i=1}^{n} \xi_k$$

$$y_k(\langle x_k, \theta' \rangle + \theta_0) \geq 1 - \xi_k \quad k = 1, \dots, n$$

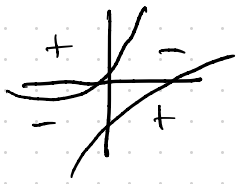
$\min \|\theta'\|$ → maximizes the margin

$\min \sum \xi_k$ → minimizes loss

C → controls trade-off.

Benifits of SVM: Most data is <u>not</u> linearly separable.

Problem with SVM: Data might have nonlinear structure.



Solution: add a kernel function

$$k(x, y) = \langle \phi(x_k), \phi(x_\ell) \rangle$$

to reduce non-linear to linear problems

Dual SVM

Algorithm

1. Input :
   - Training data $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^D \times \{\pm 1\}$
   - Kernel map $k(x, y)$
   - Regularization parameter C.

2. Output : $f : \mathbb{R}^D \to \{\pm 1\}$ of the form

$$f(x) = \operatorname{sgn}\left(\langle \theta, \phi(x) \rangle + \theta_0 \right)$$

3. Compute Kernel matrix $G = \left(k(x_i, x_j) \right)_{i,j=1}^{n}$

4. Solve Dual SVM problem by finding $\alpha \in \mathbb{R}^n$

$$\max_{\alpha \in \mathbb{R}^n} \left\{ -u^T G u + \sum_{k=1}^{n} \alpha_k \;\middle|\; \sum_{k=1}^{n} \alpha_k y_k = 0, \; 0 < \alpha_i \leq C \right\}$$

where $u = \left(\frac{1}{2} \alpha_k y_k \right)_{i=1}^{n}$

5. Define $\theta(x) = \frac{1}{2} \sum_{i=1}^{n} y_i \alpha_i k(x_i, x)$

6. Set $b = \text{median}\{ y_k - \psi(x_k) \mid \alpha_k \neq 0 \}$

7. Return $f(x) = \text{sgn}(\psi(x) + b)$.

Notebook 5 → classification

1. We'll use kernel map $k(x,y) = \langle x, y \rangle$ the standard inner product

4. ✱ JuMP → package for model

✱ NLopt → non linear optimization

algorithm: AUGLAG = "Augmented Lagrangian"

✱ globally convergent augmented Lagrangian algorithm.

local_optimizer: LD_LBFGS

Low storage algorithm, minimizes memory at each step

* Maximization function looks different. We use the fact that if α maximizes a function L then α also maximizes γL, so multiply Dual vector problem by γ.

End Compares Dual SVM as above to a Neural Network

Recall activation functions:

ReLu $\sigma(z) = (\max(0, z_i))_{i=1}^{k}$

sigmoid $\sigma(z) = \left(\frac{1}{1+\exp(-z_i)}\right)_{i=1}^{k}$

Softmax $\sigma(z) = \left(\frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_k)}\right)_{i=1}^{k}$

* On Neural Networks vs. SVM

Both Can do linear + non linear classification

NN → eventually achieves higher accuracy

SVM → Faster to train, more reliable since guarantee convergence to global minimum

After Notebook 5 — Soft Margin → dual

plus why steps 5 & 6 of algorithm work.

$$\min_{\theta \in \mathbb{R}^{D+1},\ \xi \in \mathbb{R}^n} \ \max_{\alpha, \beta \in \mathbb{R}^n,\ \alpha_k, \beta_k \geq 0} \left\{ \|\theta'\|^2 + C \sum_{k=1}^{n} \xi_k - \sum_{k=1}^{n} \beta_k \xi_k - \sum_{k=1}^{n} \alpha_k \left[y_k (\langle \theta', x_k \rangle + \theta_0) - (1 - \xi_k) \right] \right\}$$

= soft margin over $y_k (\langle \theta', x_k \rangle + \theta_0) - (1 - \xi_k) \geq 0$

↑ this is called the Lagrange function

$$\mathcal{L}(\theta', \theta_0, \xi, \alpha, \beta)$$

for soft margin SVMs.

KKT = Karush–Kuhn–Tucker conditions

Karush: 1939

Kuhn–Tucker: 1951

KKT $\Rightarrow$ min max problem above is solved when

$$\frac{\partial \mathcal{L}}{\partial \theta'} = \frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{\partial \mathcal{L}}{\partial \xi} = 0.$$

* The "kernel trick" comes from the fact that the conditions for these 3 conditions to be 0 are <u>independent</u> of θ and ξ, so we can first solve maximum function!

$$\frac{\partial \mathcal{L}}{\partial \theta'} = 2\theta' - \sum_{k=1}^{n} \alpha_k y_k x_k = 0$$

$$\theta_*' = \frac{1}{2} \sum_{k=1}^{n} \alpha_k y_k x_k$$

For algorithm step 5 $y(x) = \langle \theta_*', x \rangle$ = portion of x in the optimal direction.

Rest of KKT gives [6] of algorithm

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = -\sum_{k=1}^{n} \alpha_k y_k = 0$$

$$\Rightarrow \sum_{k=1}^{n} \alpha_k y_k = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi} = \Big(C - (\alpha + \beta) \Big)_{k=1}^{n} = 0$$

$$\Rightarrow C = \alpha + \beta = 0 \leq \alpha_k \leq C.$$

Plugging back into $\mathcal{L}$ the fact that $\alpha + \beta = C$, $\sum \alpha_k y_k = 0$

$$\mathcal{L} = \|\theta'\| - \sum_{k=1}^{n} \alpha_k \left(y_k \left(\langle \theta', x_k \rangle + \theta_0 \right) - 1 \right)$$

Maximizing over α means $\left(y_k \langle \theta', x_k \rangle + \theta_0 \right) - 1 \leq 0$

$$\Rightarrow \alpha_k = 0$$

$$\mathcal{L} = \|a\|^2 - \sum_{\alpha_k > 0} \alpha_k \left(y_k \langle \theta' + x_k \rangle + \theta_0 \right) - 1 \Big)$$

minimizing $\mathcal{L}$ over $\theta \Rightarrow$ maximize $\alpha_k = C$

$$\mathcal{L} = \|a\|^2 - C \sum_{\alpha_k > 0} \underbrace{y_k(\langle \theta', x_k \rangle + \theta_0) - 1)}_{\geq 0}$$

So θ_0^* minimizes $\sum_{\alpha_k > 0} \left| 1 - y_k(\langle \theta_*', x_k \rangle + \theta_0) \right|$

Multiplying by $(y_k) = 1$, and $y_k^2 = 1$

θ_0^* minimizes $\sum_{\alpha_k > 0} \left| y_k - \langle \theta_*', x_k \rangle - \theta_0 \right|$

By exercise 3.7 minimizer is

$$\text{median}_{\substack{1 \leq k \leq n \\ \alpha_k > 0}} \{ \underbrace{y_k - \langle \theta_*', x_k \rangle}_{\| \; y_k - \psi(x_k)} \}$$