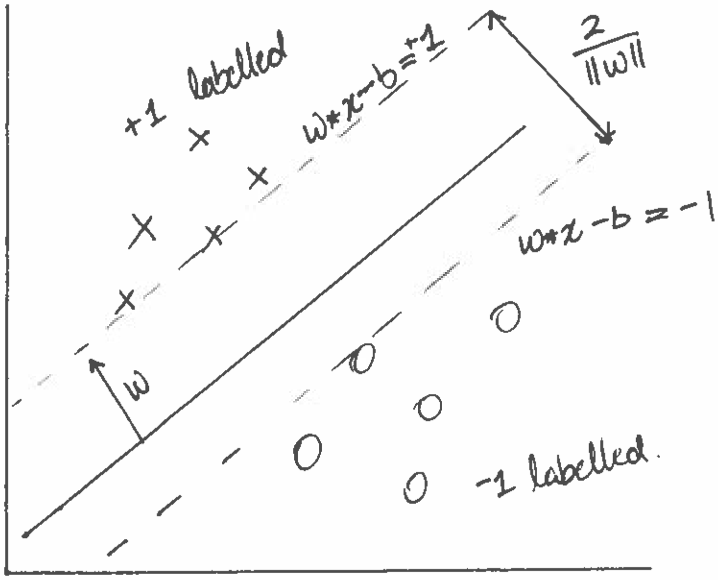
SVM

Training data: $(\vec{x}_i, y_i)$; $i = 1 \ldots n$



Linearly separable data

Anything above $\vec{w}^T\vec{x} - b = 1$ is a "x" datapoint

Maximize $\frac{2}{\|\vec{w}\|} \Rightarrow$ Minimize $\|w\|_2$

$\Rightarrow$ If $y_i = 1$, $\vec{w}^T\vec{x}_i - b \geq 1$
$y_i = -1$, $\vec{w}^T\vec{x}_i - b \leq -1$.
} Points must be correctly classified

$\Rightarrow y_i(\vec{w}^T\vec{x}_i - b) \geq 1 \quad \forall \; 1 \leq i \leq n$ — Constraint

Optimization problem:

$$\min_{w,b} \|\vec{w}\|_2^2$$

Quadratic optimization, solve w/ Lagrange multipliers.
↳ Quadratic $\Rightarrow$ $\boxed{\text{single global minima.}}$

If we remove support vectors, hyperplane changes!

sub. ~~$y_i(\vec{x}$~~ $y_i(\vec{w}^T\vec{x}_i - b) \geq 1 \quad \forall i \in \{1 \ldots n\}$.

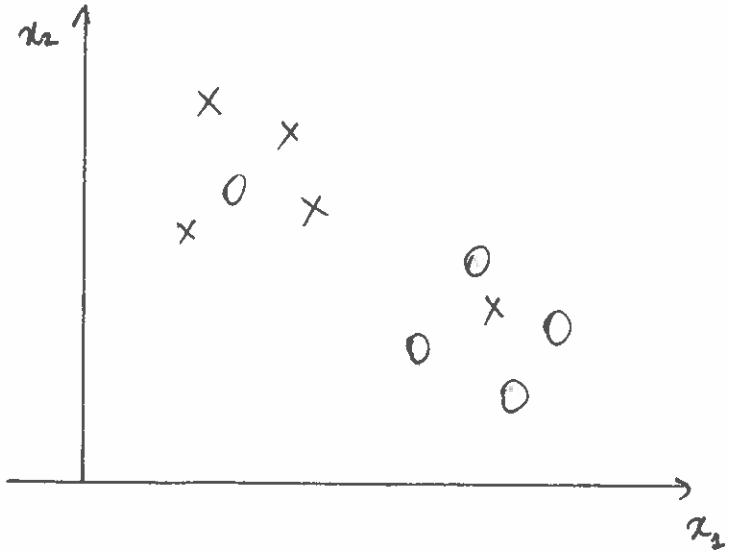
↖ hyperplane is determined completely by nearest x_i on either side!
$\Rightarrow$ Most difficult pts to classify

Use the optimization of maximizing margin to reduce number of nonzero weights ~~that~~ $\Rightarrow$ only look at weights that matter $\Rightarrow$ ie. correspond to the support vectors!

Key diff b/w SVM & $\underbrace{\text{NNs/linear reg}}_{\text{use all data!}}$

Ultimately, classifier: sign$(\vec{w}^T\vec{x} - b)$ for some new x, using learned/optimized $\vec{w}$, b.

"Fuzzy" data



Some points' labels are "misclassified"

Modify minimization: (using hinge loss)

$$\max\left(0,\ 1 - y_i(\vec{w}^T\vec{x}_i - b)\right)$$

returns 0 if point is correctly classified/labelled

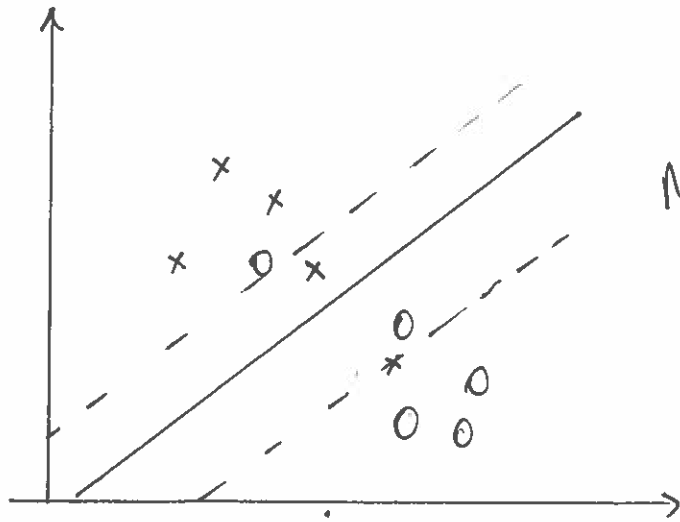
(0,1), proportional to distance from margin if incorrectly classified.

$$\Rightarrow \text{minimize} \quad \lambda\|\vec{w}\|_2^2 + \left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0,\ 1 - y_i(\vec{w}^T\vec{x}_i - b)\right)\right]$$

Rewrite: $\displaystyle\min_{w,b,\zeta} \|\vec{w}\|_2^2 + C\sum_{i=1}^{n} \zeta_i$ Riemann zeta function

sub. $y_i(\vec{w}^T\vec{x}_i - b) \geq 1 - \zeta_i,\quad \zeta_i \geq 0 \quad \forall\, i \in \{1 \dots n\}$

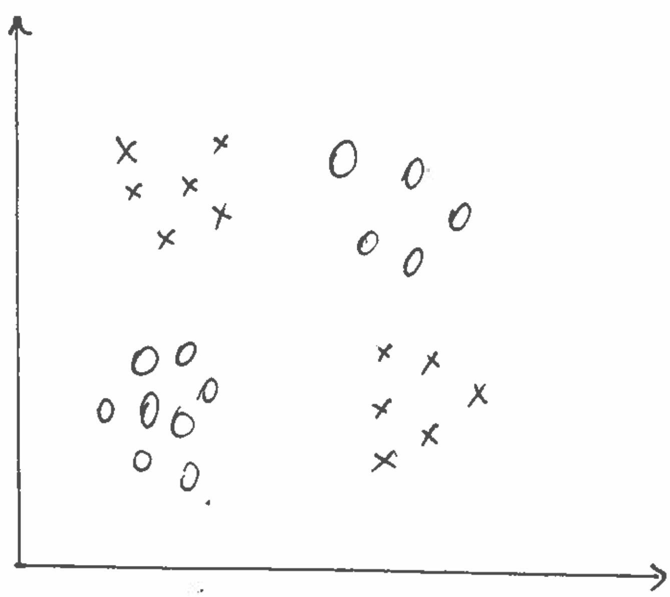
⇒ Our learned classifier now looks like:



Number of support vectors = 4

Very similar to linear discriminant analysis (LDA)

Non-linearly Separable Data:



XOR dataset

Gain linear separation by mapping data into a high-dimensional space

We actually solve SVM optimization via the dual formulation (removes dependence on $\vec{w}$, b)

Dual problem -

Interchange variables & constraints:

$$\begin{array}{ll} \min & c \circ x \\ \text{sub.} & A \circ x = b \\ & x \geq 0. \end{array} \longrightarrow \begin{array}{ll} \max & b \circ y \\ \text{sub.} & A^T \circ y \leq c. \\ & y \text{ Free. (Unconst.)} \end{array}$$

Here:

$$\begin{array}{ll} \min & \|w\|_2^2 \\ \text{sub.} & y_i \cdot (\vec{w}^T \vec{x}_i - b) \ldots \end{array} \longrightarrow \boxed{\begin{array}{ll} \min. & f(\vec{w}) \\ \text{sub.} & g_j(\vec{w}) \leq 0. \\ & h_k(\vec{w}) = 0 \end{array}}$$

General form (Lin sep. / fuzzy).

$\downarrow$ Dual formulation

$$\mathcal{L} \equiv \frac{1}{2} f(\vec{w}) + \sum_j a_j g_j(\vec{w}) + \cancel{\sum_k \beta_k h_k(\vec{w})}$$

$0 \because$ no equality const.

(Method of Lagrange multipliers)

$\Rightarrow$ For a min / max:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0 \quad ; \quad \frac{\partial \mathcal{L}}{\partial \beta_k} = 0$$

From strong duality & Kuhn - Tucker thms.,

$$\begin{array}{ll} \max. & \mathcal{L}(\alpha, \beta). \\ \text{sub.} & \alpha_j \geq 0 \quad \forall j. \end{array} \equiv \text{original primal form.}$$

Generally true $\forall$ convex fns. $f(w)$.

Further, if $\hat{w}$ is optimal soln of primal & $\hat{a}$ & $\hat{\beta}$ are optimal solns of dual:

$$\left. \begin{array}{l} f(\hat{w}) = \mathcal{L}(\hat{a}, \hat{b}) \\ \hat{a}_j g_j(\hat{w}) = 0 \quad \forall j \end{array} \right\} \text{Karush-~~Kuhn~~ Kuhn-Tucker (KKT) complementarity condition.}$$

If we sub. our original constraints for the SVM problem into the Lagrangian,

$$\mathcal{L} = \frac{1}{2}\vec{w}\cdot\vec{w} + \sum_i \alpha_i \left[1 - y_i(\vec{w}\cdot\vec{x}_i + b)\right]$$

For max/min:

$$0 \overset{set}{=} \frac{\partial \mathcal{L}}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i \qquad \text{and} \qquad 0 = \frac{\partial \mathcal{L}}{\partial b} = \sum_i \alpha_i y_i$$

$$\Rightarrow \hat{w} = \sum_i \hat{\alpha}_i y_i \vec{x}_i$$

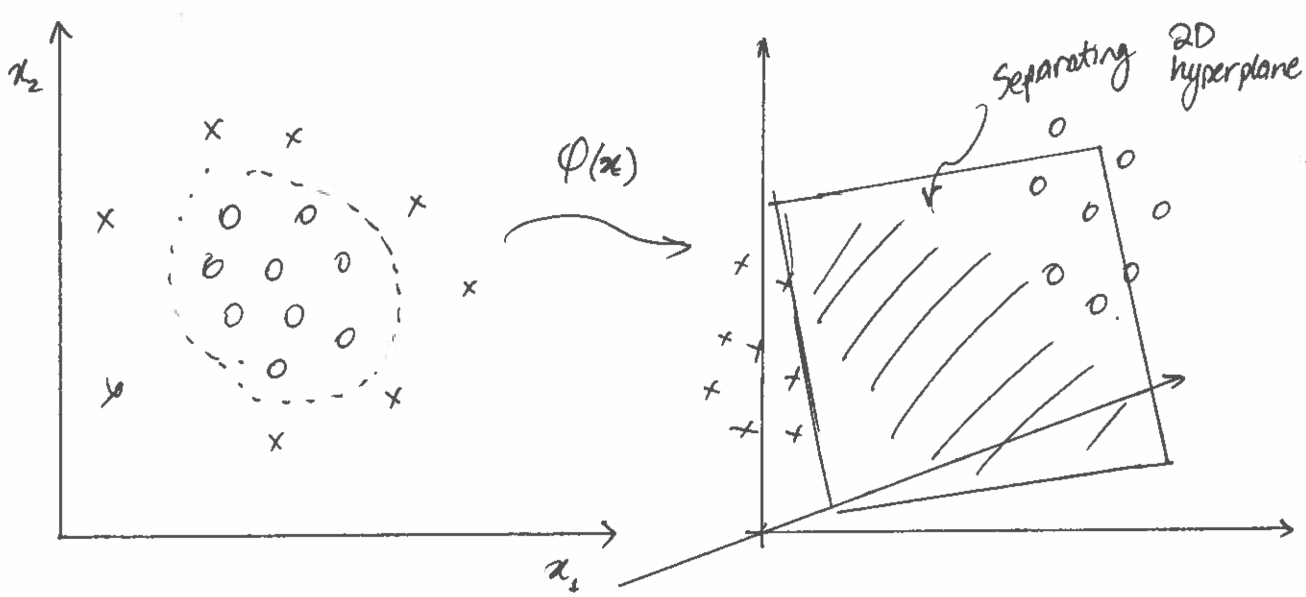
$$\Rightarrow \mathcal{L}(\alpha) = \sum_j \alpha_j - \frac{1}{2}\sum_{j,k} \alpha_j y_j (\vec{x}_j \cdot \vec{x}_k) y_k \alpha_k$$

The only place where $\vec{x}$ shows up is here!

This is the only thing that scales w/ number of features. ⇒ all other parts scale w/ num. pts. ⇒ Primal to dual → shift scaling from number (primal) of features to number of pts (dual!)

⇒ Favors data w/ ↓ numbers of data pts., but huge features

Now, lets talk about non linearity. ⇒ "Kernel Trick".



Primal form: $\min . \frac{1}{2}\vec{W}\cdot\vec{W} + \lambda\sum_i \Xi_i$

sub. $y_i(\vec{W}\cdot\varphi(\vec{x}_i) + B) \geq 1 - \Xi$ $\quad i = 1 \ldots m.$

But, embedding dim may be $\mathbb{R}^{\text{Billions}}$!.

$\Longrightarrow$ Intractable!

Dual problem: ~~min $\frac{1}{2}\alpha \cdot \mathrm{diag}(y$~~

$\min . -\sum_j \alpha_j + \frac{1}{2}\sum_{j,k} \alpha_j y_j K_{jk} y_k \alpha_k$

$K_{jk} = K(x_j, x_k) = \varphi(x_i) \cdot \varphi(x_k)$ dot prod.

But, we don't need to explicitly know mapping $\varphi(x)$!

All we need is some way to compute K_{jk} that could have come from some $\varphi(x)$! $\Rightarrow$ $m \times m$ matrix w/ mathematical properties of inner product!

- $K_{ij} = K(x_i, x_j)$ must be symm. in i, j, and nonneg. eigenval.
- K can be composed by addition, mult., scaling by const.

Popular Kernels:

linear: $K(x_i, x_j) = x_i \cdot x_j$.

power: $K(x_i, x_j) = (x_i \cdot x_j)^d$ $\quad 2 \leq d \leq 20$ (usually).

polynom: $K(x_i, x_j) = (a x_i \cdot x_j + b)^d$

sigmoid: $K(x_i, x_j) = \tanh(a x_i \cdot x_j + b)$.

Gaussian rbf: $K(x_i, x_j) = e^{-\frac{1}{2}\frac{|x_i - x_j|^2}{\sigma^2}}$

Tips for using kernels:

- Gaussian rbf is very popular, ∵ only 1 hyperparam.

  Guess good initial σ via avg. distance b/w points in <u>feature</u> <u>space</u>.

- Polynomial kernels:

  Start by choosing a, b s.t. $a x_i \cdot x_j + b$ b/w $-1, 1$ $\forall i, j$.

  d ⇒ roughly interpret as number of features to be mixed ~~to~~ during partitioning.

  ⇒ $d=1$ ⇒ partition space by 1 feature at a time

  2 ⇒ 2 features ...

Diff b/w power & polynomial: power considers only d features at once, polynomial considers all combos of d or fewer features.