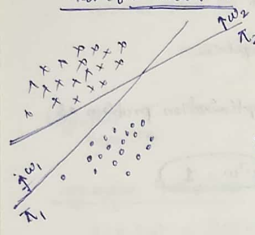
SVM (Support Vector Machines).

Geometric Intuition



There will be many possible hyperplanes that separate positive (+)ve pts from (−)ve pts.

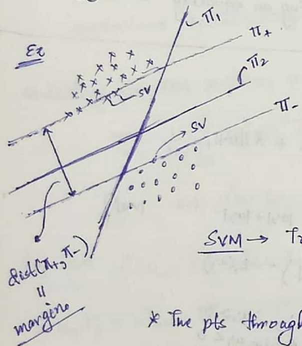
We had learnt in LR that the probability of a point belonging to any class given it is very close to the hyperplane will be close to $\boxed{0.5}$

So, we want a hyperplane that separates (+)ve pts and (−)ve pts as far away as possible.

Key idea of SVM

such a hyperplane is called margin-maximizing hyperplane.

Ex



π_2 better than π_1.

$\pi_+ \rightarrow$ parallel to π & touches the closest (+)ve point to π.

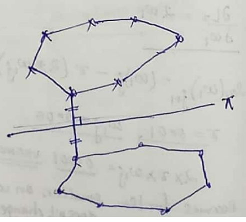
$\pi_- \rightarrow$ parallel to π & touches the closest (−)ve point to π.

SVM $\rightarrow$ Try to find a π that maximizes the margin $(dist(\pi_+, \pi_-))$.

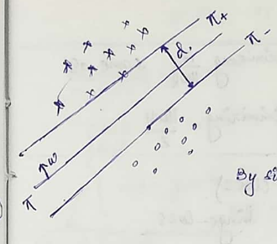
* The pts through which π_+ & π_- pass through are called support vectors.

Alternative geometric formulation of SVM

① Perform convex-hull of (+)ve & (−)ve pts.
② find the shortest line connecting these hulls.
③ Bisect it.



Mathematical formulation



We need to find a hyperplane

π :- margin maximizing.

π: $w^T x + b = 0$ (w not necessarily unit vector).

π_+: $w^T x + b = 1$

π_-: $w^T x + b = -1$

By simple co-ordinate concepts we can get

$$d = \frac{2}{\|w\|_2}$$

So, we have to maximize 'd' with the constraint that all (+)ve pts lie above π_+ & all (−)ve pts below π_-.

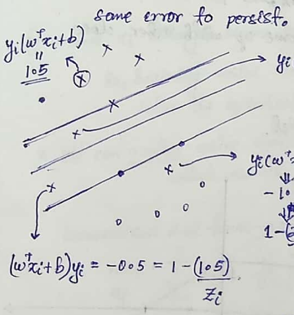
So, the optimization function becomes

constraint optimizⁿ problem

$$\left.\begin{aligned} (w^*, b^*) &= \underset{(w,b)}{\text{argmax}} \ \frac{2}{\|w\|} \quad \text{— margin} \\ \text{s.t.}\ \ & y_i(w^T x_i + b) \geq 1 \ \forall \ \underline{x_i \in D_{train}} \\ & \qquad\qquad\qquad\qquad\ \ y_i \end{aligned}\right\} \text{— Hard-margin SVM constraint}$$

But what if we have a dataset that's not linearly separable but almost linearly separable?

↳ We formulate an alternative soft-margin constraint that allows some error to persist.


$y_i(w^T x_i + b) \shortparallel 1.5$
$y_i(w^T x_i + b) = 0.5 = 1 - \underset{z_i}{(0.5)}$
$y_i(w^T x_i + b) \Downarrow -1.5 \Downarrow 1 - \underset{z_i}{(2.5)}$
$(w^T x_i + b) y_i = -0.5 = 1 - \underset{z_i}{(1.5)}$

We define a new metric z_i for all pts such that

$$z_i = \begin{cases} 0, & \text{for correctly classified i.e } y_i(w^T x_i + b) \geq 1. \\ z_i > 0 \ \& \ \text{~~[illegible]~~ proportional to dist away fm the correct hyperplane } (\pi_+ \text{ or } \pi_-), & \text{for incorrectly classified pts.} \end{cases}$$

correctly & incorrectly wrt π_+ & π_- considered here.

This can be written in form of eqⁿ as

$$\boxed{\begin{gathered} y_i(w^T x_i + b) \geq 1 - z_i \\ z_i \geq 0 \end{gathered}}$$

Now, we want to minimize these errors.

So, our optimization function becomes

$$\underset{w,b}{\arg\min}\ \frac{\|w\|}{2} + c\,\frac{1}{n}\sum_{i=1}^{n}\zeta_i$$

$\frac{\|w\|}{2}$ → $\frac{1}{\text{margin}}$ → (regulrzn)

$\frac{1}{n}\sum_{i=1}^{n}\zeta_i$ ↳ avg dist of misclassified pts. ↳ (loss)

maximizing $\frac{2}{\|w\|}$ same as minimizing $\frac{\|w\|}{2}$

s.t. $y_i(w^T x_i + b) \geq 1 - \zeta_i \quad \forall i$ hinge-loss

$\zeta_i \geq 0$

def^n of ζ_i

— This is soft margin formulation of SVM.

$c \rightarrow$ hyper-parameter.

$c\uparrow \Rightarrow$ tendency to make mistakes $\downarrow$ on D_{train} $\Rightarrow$ overfit $\Rightarrow$ high variance.

$c\downarrow \Rightarrow$ underfit $\Rightarrow$ high bias.

$$\boxed{c = \frac{1}{\lambda}}$$

Loss-minimization interpretation

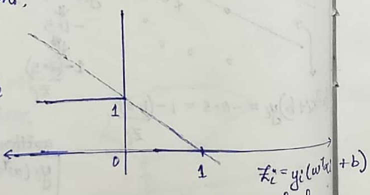
We earlier saw that 0_1 loss is ideal. ~~[illegible]~~ loss function.
But due to it being non-differentiable, we come up with other close approximate loss functions.

logistic loss gave $\longrightarrow$ Logistic regression model.

squared loss gave $\longrightarrow$ linear regression.

hinge loss gives $\longrightarrow$ SVM.

* Although hinge loss also isn't diffentiable at $\zeta_i = 1$, but since it is continuous unlike 0_1 loss, we can work around hacks to work with it.



$\zeta_i > 0$ – x_i correctly classified.
$\zeta_i < 0$ – x_i incorrectly classified.

hinge-loss :– 0, $\zeta_i \geq 1$.
$1 - \zeta_i$, $\zeta_i < 1$

Alternatively,
hinge-loss = $\max(0, 1-\zeta_i)$

Dual form of SVM

$$\begin{cases} \min\limits_{w,b}\ \frac{1}{2}\|w\| + c\sum\limits_{i=1}^{n}\zeta_i \\ \text{s.t. } y_i(w^T x_i + b) \geq 1 - \zeta_i\ \forall i \\ \quad \& \ \zeta_i \geq 0. \end{cases}$$

Primal form of Soft-margin SVM.

$$\max_{\alpha_i}\ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\, x_i^T x_j$$

s.t. $\alpha_i \geq 0$

& $\sum_{i=1}^{n}\alpha_i y_i = 0$

Dual form of Soft margin SVM.

Important observations from dual-form formulation.

① for every x_i, we have an α_i

② x_i's only occur in the form of $x_i^T x_j$.

On solving we get the model as

$$\boxed{f(x_q) = \sum_{i=1}^{n}\alpha_i y_i\, x_i^T x_q + b}$$

— for non SV, $\alpha_i = 0$.
So, for $f(x_q) \rightarrow$ only the support vectors matter.

③ $\alpha_i > 0$ only for support vectors, else 0.

④ – since x_i always occurs only as $x_i^T x_j$, $\Rightarrow x_i \cdot x_j \Rightarrow$ cosine_sim(x_i, x_j).
So, basically cosine similarity values are only required to solve the optimizat^n & ultimately get the model.

* We can replace cosine_similarity with any other kind of similarity. which makes it much more powerful.

Generalized dual form →

$$\max_{\alpha_i}\ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\, K(x_i, x_j)$$

$K(x_i, x_j)$ — called kernel function. can be any kind of similarity b/w x_i & x_j.

Even during evaluation x_i occurs only as $x_i^T x_q$

$$f(x) = \sum_{i=1}^{n}\alpha_i y_i\, \underbrace{x_i^T x_q}_{K(x_i, x_q)} + b$$

So, we can use similarity values for even prediction as well.

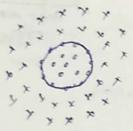
Kernel trick

Intuitively if we see, SVM & LR are very similar.

* The most important idea of SVM is the kernel trick.

Kernel trick → replacing $x_i^T x_j$ with a generalized similarity function $K(x_i, x_j)$

This gives SVM potential to perform in non-linear datasets.



linear-SVM — fail.

log-regs^n — fail.

log-reg^n + feature transform — succeed.

kernel SVM + right kernel — succeed.

Just like in feature transformat^n, we map data pts to a new space, similarly, the kernel in SVM does the mapping & then tries to find a ~~plan~~ hyperplane in that mapped ~~[illegible]~~ space.

Polynomial-Kernel

Given circular data like above, we can

① handle using feature transforms.

$$(f_1, f_2) \xrightarrow{F.T.} (f_1^2, f_2^2) —$$

② fit right kernel.

General polynomial kernel of form –

$$\boxed{K(x_1, x_2) = (x_1^T x_2 + c)^d}$$

A quadratic kernel can be written as.

$$K(x_1, x_2) = (x_1^T x_2 + 1)^2 \qquad \text{assuming } c = 1.$$

let $x_1 \langle x_{11}, x_{12} \rangle$ & $x_2 \langle x_{21}, x_{22} \rangle$

$$K(x_1, x_2) = (1 + x_{11}x_{21} + x_{12}x_{22})^2$$

$$= 1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{21}x_{12}x_{22}$$

Say $\langle 1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12} \rangle$ — x_1'

$\langle 1, x_{21}^2, x_{22}^2, \sqrt{2}x_{21}, \sqrt{2}x_{22}, \sqrt{2}x_{21}x_{22} \rangle$ — x_2'

Then the product we have got due to quadratic kernel is equivalent to

$x_1'^T x_2'$.

* So, the kernel basically internally 1st maps x_1 & x_2 to x_1' & x_2' and then finds their similarity (cosine).

kernelization → $d \xrightarrow[\text{internally}]{\text{feature transform}} d'$ $(d' > d)$

* In our example we went fm 2-D data to 6-D data.

That's what Mercer's theorem says.

It says Kernel trick is basically nothing but

$$d \xrightarrow[\text{transform}]{\text{implicit}} d' \quad (d' > d).$$

└ data linearly separable in this space.

The challenge is to select the right kernel.

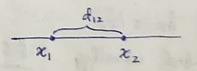
RBF-Kernel (Radial Basis Function).

↓

most popular and general purpose kernel.

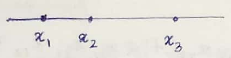
$$\boxed{K_{rbf}(x_1, x_2) = \exp\left\{\frac{-\|x_1 - x_2\|^2}{2\sigma^2}\right\}}$$

σ — hyperparameter.



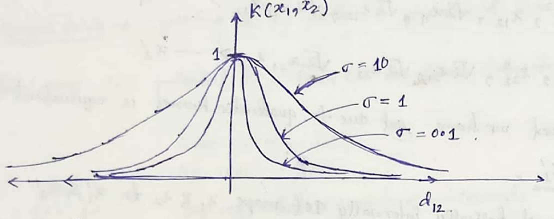
$$\boxed{K(x_1, x_2) = \frac{1}{e^{d_{12}^2/2\sigma^2}}}$$

$d_{12} \uparrow$, $K(x_1, x_2) \downarrow$.



$K_{RBF}(x_1, x_2) > K_{RBF}(x_1, x_3)$

Impact of σ



for ~~large~~ small σ, the kernel value (similarity value) decreases very quickly with increasing distance.

Whereas for large σ, the kernel value decreases gradually, slowly.

So, small σ is similar to small 'k' in KNN. – considering very few closest neighbours.

large σ ⇒ large k → considering more neighbours which are even a bit farther.

K-NN works pretty good but its biggest drawback is runtime complexity
↳ needs to store all the datapoints.
or if optimised, LSH.

for RBF-SVM, we just need to store α_i's of SVs.
That's it.
And usually, #SVs << n

If can't decide which kernel, go with RBF.
2 hyperparameters – c, σ. —— Random search.

Domain Specific Kernels

* With research over the years we have come up with domain specific kernels.
Ex → String kernels for text classification,
genome kernels (Bioinformatics),
graph kernels (for graph-data).

Domain Knowledge + Appropriate feature transformations ⟶ Custom-domain specific kernels.

Train and Runtime Complexity

Train :– SGD (Stochastic Gradient Descent).
Specialised algo to solve dual form → sequential minimal optimization. (SMO).

libSVM → best library for training SVM.
(c/c++)

Training time :– $O(n^2)$ for Kernel SVMs.
(2007) :– $O(nd^2)$ if $d<n$

If n is large, $O(n^2)\uparrow\uparrow$
So, typically don't use SVM when n is large.

Runtime :→ $f(x_q) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_q) + b.$
↑ depends on
$\alpha_i = 0$ for non-SV

#SVs = k (say).
Then runtime complexity = $O(kd)$ $1 \le k \le n$

logistic regression · runtime – $O(d)$

In normal SVM we have no control over no. of support vectors.

nu-SVM

C-SVM - original formulation. c - hyperparameter $c \geq 0$

nu-SVM - alternative formulation. nu- hyperparameter $0 \leq nu \leq 1$

nu $\geq$ fraction of errors.

nu $\leq$ fraction of SVs.

Ex- If we want to ~~have~~ permit an error of 10% in worst case,

nu = 0.01

Also, #SVs is greater than 10% of n.

We can have an upper bound on error, but not on #SVs.

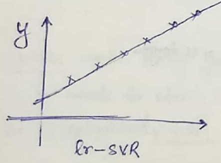
Support Vector Regression (SVR)

Math:- $\min_{w,b} \frac{1}{2} ||w||^2$ s.to $y_i - (w^T x_i + b) \leq \epsilon$

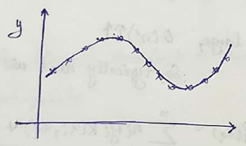
& $(w^T x_i + b) - y_i \leq \epsilon$

$\epsilon \geq 0$ $\downarrow$ hyper-parameter.

This can also be kernelized.



lr-SVR



kernel-SVR.

$\epsilon\downarrow$ $\Rightarrow$ errors low on training data $\Rightarrow$ overfit $\uparrow$ $\Rightarrow$ high variance

$\epsilon\uparrow$ $\Rightarrow$ errors on D_{train} higher $\Rightarrow$ underfit $\Rightarrow$ high bias.

Real world cases

feature engg & transformation $\rightarrow$ kernel design, finding the right kernel.

decision surface $\rightarrow$ lr. SVM $\rightarrow$ hyperplane.

Kernel SVM $\rightarrow$ d x_i $\rightarrow$ non-linear surface

$\downarrow$

d' x_i' $\longrightarrow$ linear surface.

Similarity matrix ✓

Interpretability and feature importance -

for lr. SVM, same as L.R.

for kernel SVM, difficult.

Outliers $\rightarrow$ generally v. little impact.

exception ** RBF with small σ = KNN with small k. -

Bias-variance $\rightarrow$ $c\uparrow$ $\Rightarrow$ overfit

$c\downarrow$ $\Rightarrow$ underfit.

large d $\rightarrow$ v. good for SVMs. as SVMs eventually map to higher dimensions only.

Best case $\rightarrow$ have right kernel ✓

worst case $\rightarrow$ n is large. $\longrightarrow$ high train time

#SVs large $\longrightarrow$ low latency not possible.