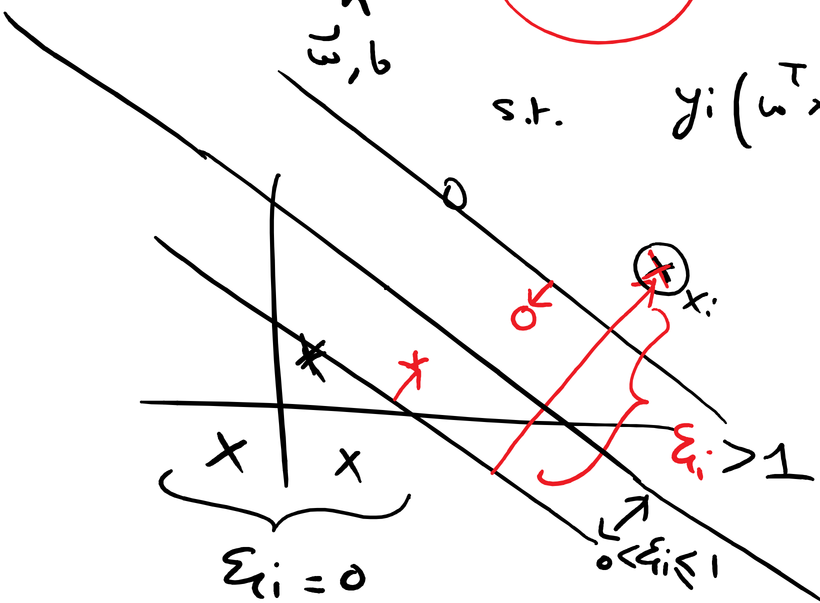
SVM objective

margin — Penalty / Violations

$$\min_{\vec{w},b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i^k$$

s.t. $\quad y_i(w^T x_i + b) \geq 1 - \xi_i$



Dual Objective

α_i : one for each point

$0 \leq \alpha_i \leq C$

$$\boxed{\max_{\alpha_i} \left\{ \sum \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i x_j) \right\}}$$

$$\boxed{\vec{w} = \sum_{\alpha_i > 0} \alpha_i y_i \phi(x_i)}$$

linear

$$\vec{w} = \sum_{\alpha_i > 0} \alpha_i y_i x_i$$

we ~~cannot~~ / do not here
access to $\vec{w}$

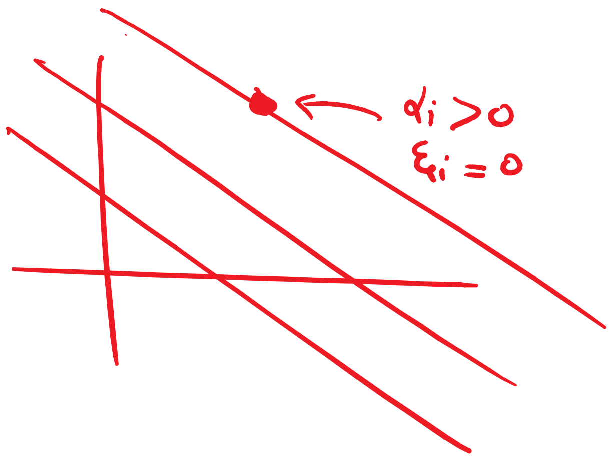
bias: b?

$$\boxed{\alpha_i\left(\underbrace{y_i\left(w^T x_i + b\right) - 1 + \xi_i}\right) = 0}$$

for support vectors

$C > \alpha_i > 0 \implies y_i\left(w^T x_i + b\right) - 1 + \xi_i = 0$

$\hookrightarrow \xi_i = 0$



$$y_i\left(w^T x_i + b\right) = 1$$

$\Downarrow$ In kernel space

$$\boxed{y_i\left(w^T \phi(x_i) + b\right) = 1}$$

$$y_i b = 1 - y_i w^T \phi(x_i)$$

$$w = \sum_{\alpha_j > 0} \alpha_j y_j \phi(x_j)$$

$$\boxed{b = \frac{1}{y_i} - w^T \phi(x_i)}$$

$$b = \frac{1}{y_i} - \left(\left(\sum \alpha_j y_j \phi(x_j)\right)^T \phi(x_i)\right)$$

$C > \alpha_i > 0$

original support

$$\boxed{b = \frac{1}{y_i} - \sum_{\alpha_j > 0} \alpha_j y_j K(x_j, x_i)}$$

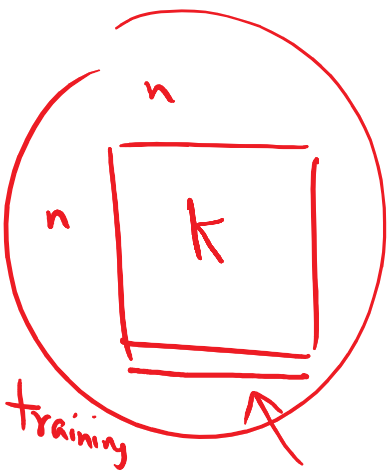
Original support vectors

$$h(\vec{z}) = w^T \phi(z) + b$$

$$= \sum_{\alpha_i > 0} \alpha_i y_i K(x_i, z) + b$$

solved for

z is a new test case



$$\boxed{svm(\vec{z}) = sign(h(\vec{z}))}$$

z

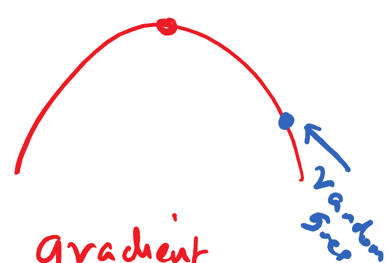
$K(x_i, z)$

$$\max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$\boxed{0 \le \alpha_i \le C}$ ← Constraint

$$\vec{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}$$

n - values to solve for



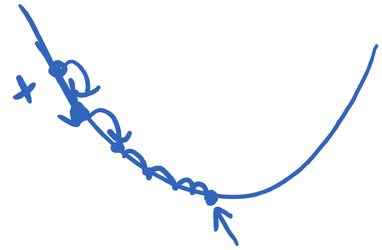
gradient ascent or hill climbing

Start at some solution $\vec{\alpha}_0$, compute the gradient

α_0, compute the gradient

derivative at $\vec{\alpha}_0$

∇_{α_0}



$$\boxed{\vec{\alpha}_{i+1} = \vec{\alpha}_i + \eta \nabla_{\vec{\alpha}_i}}$$

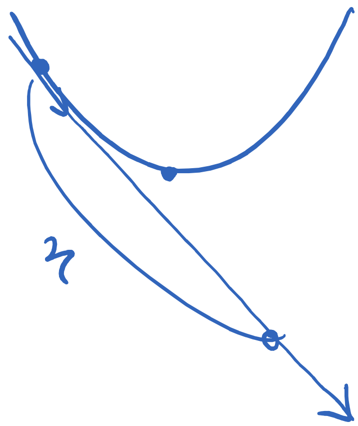
Step size

steepest improvement

SVM dual objective

convex quadratic program

$\Rightarrow$ unique global optimal solution

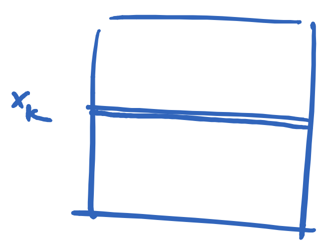


$\alpha_1 + \alpha_2 + \dots + \alpha_k + \dots$

$$J = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i x_j)$$

$$\nabla_k = \frac{\partial J}{\partial \alpha_k} = \boxed{1 - y_k \left(\sum_{i=1}^{n} \alpha_i y_i k(x_i x_k) \right)}$$

Compute gradient for k-th element of $\vec{\alpha}$



$\vec{\alpha}_0 = (0, 0, \dots, 0)$

$\vec{\alpha}_0 = (0, 0, \ldots, 0)$

repeat until convergence

$\vec{\alpha}_{prev} = \alpha$

for $k=1$ to n ← randomize choice of k

update the k-th component

$\alpha_k = \underbrace{\alpha_k}_{old} + \eta \cdot \underbrace{\nabla_k}_{gradient}$

↑ Step Size

$$\boxed{\alpha_k = \alpha_k + \eta \cdot \left(1 - y_k \left(\sum_{i=1}^{n} \alpha_i y_i \, k(x_i \cdot x_k)\right)\right)}$$

Clipping: $0 < \alpha_k \leq C$ ← ensure

old

α_{new}

1) Approach 1: batch approach

new values are not used until the next iteration

2) SGD: Stochastic Gradient Descent

Using new updated values the moment they become available

much faster convergence

for very large data each step is cheap & can be done completely in parallel

Convergence: monitor α's:

$\| \vec{\alpha}_{prev} - \alpha_{new} \| \leq \varepsilon$ Stop

↑ Convergence threshold

Classification Evaluation

D ← sample of labeled data

$$(\vec{x}_i, y_i)_{i=1}^{n}$$

point

true class

SVM
DT
KNN
NB

$$M(\vec{z}) = \hat{y}$$

model / classifier

test case

prediction

D

D_{train} 80%

in sample

D_{test} 20%

out of sample

truly blind

$\vec{z}$, true class y

$\hat{y}$ vs y

SVM
→ C? ← how to choose C

KNN
→ k ← # of neighbors to select..

D_{train} 90%

$D_{validation}$ 10% ← used for parameter tuning

$$\text{Accuracy} = \frac{\text{\# of correct predictions}}{|D_{test}|}$$

$z_1 \to \hat{y}_1 \quad y_1$

$z_2 \to \hat{y}_2 \quad y_2$

$|D_{test}|$

$|D_{test}| = N \qquad = \dfrac{\#\{\hat{y}_i = y_i\}}{N}$

$z_1 \to \hat{y}_1 \quad y_1$

$z_2 \to \hat{y}_2 \quad y_2$

$\vdots$

$z_N \to \hat{y}_N \quad y_N$

Predicted True

Error Rate : $1 - \text{Accuracy}$

: $\dfrac{\#\{\hat{y}_i \neq y_i\}}{N}$

global measures of performance

P	N
10%	90%

D_{train}

D_{test}

$\text{Null}(\vec{z}) = N$

Null has 90% accuracy !!!

Class specific accuracy / coverage

P	N	
0%	100%	$\leftarrow$ accuracy

Confusion matrix

	True P	True N
P	n_{PP}	n_{PN}

$n_{PP} = \#\{z \to \hat{y} = P \quad y = P\}$

		True P	True N
Prediction	P	n_{PP}	n_{PN}
	N	n_{NP}	n_{NN}

TP: True positive

$z \to \hat{y} = N \quad y = N$

TN: True negative

		True P	True N
Predict	P	TP	FP
	N	FN	TN

FP: False Positive

$z \to \hat{y} = P \quad y = N$

FN: False Negative

$z \to \hat{y} = N \quad y = P$

Confusion matrix

		True P	True N	
Predict	P	TP n_{PP}	FP n_{PN}	m_P
	N	FN n_{NP}	TN n_{NN}	m_N
		n_P	n_N	n

$|D_{test}| = n$

$n_P =$ true # of P

$n_N =$ true # of N

$m_P =$ # of predicted P

$m_N =$ # of predicted N

Positive class:

Precision or accuracy for P

$$= \frac{\text{\# of correct P predictions}}{\text{\# of predictions for P}} = \frac{n_{PP}}{m_P} = \frac{TP}{TP + FP}$$

Coverage or recall for P

$$= \frac{\text{\# of correct P}}{\text{\# of P in } D_{test}} = \frac{TP}{n_P} = \frac{TP}{TP + FN}$$

Negative class

Precision: $\frac{TN}{-} = \frac{n_{NN}}{m_N}$

Precision : $\frac{TN}{TN+FN} = \frac{n_{NN}}{m_N}$

recall/coverage: $\frac{TN}{n_N} = \frac{TN}{TN+FP}$

<u>Precision vs. recall tradeoff</u>

F score$_p$: harmonic mean of Precision & recall for a class

$$F\text{score}_p : \frac{2}{\frac{1}{\text{prec}_p} + \frac{1}{\text{recall}_p}} = \boxed{\frac{2 \cdot \text{prec}_p \cdot \text{recall}_p}{\text{prec}_p + \text{recall}_p}}$$

Max = 1
min = 0