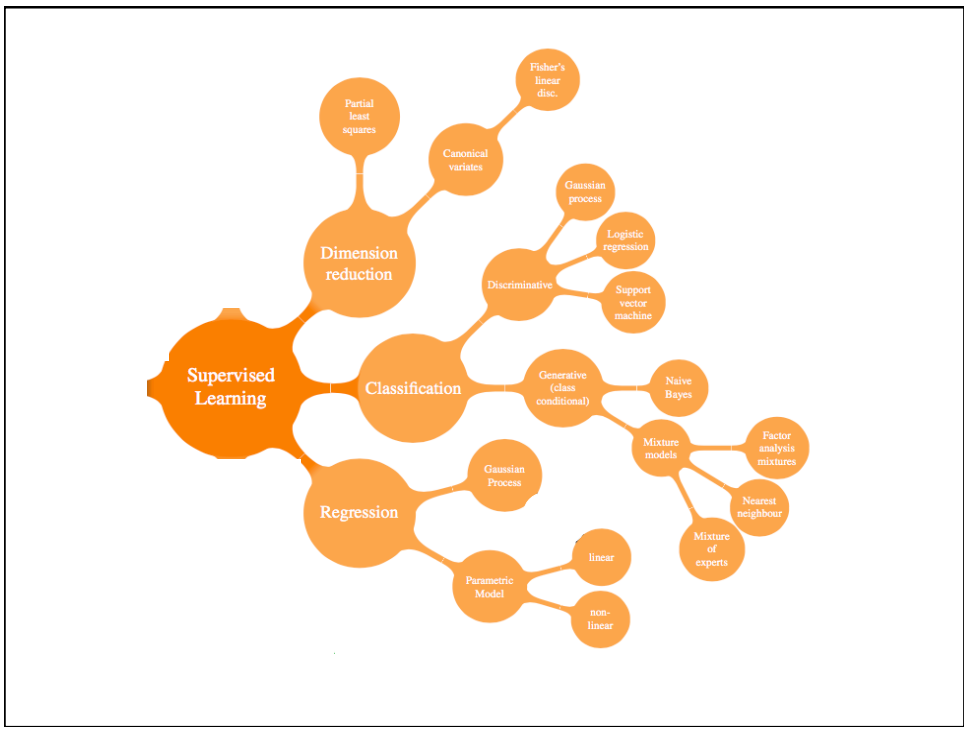


SVM

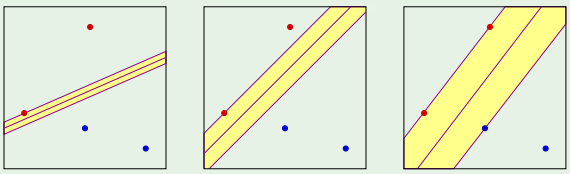
(Description)

Better linear separation

Linearly separable data

Different separating lines

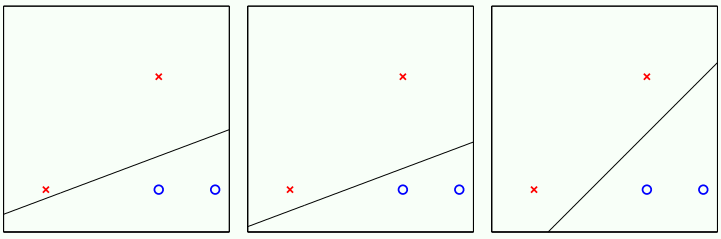
Which is best?



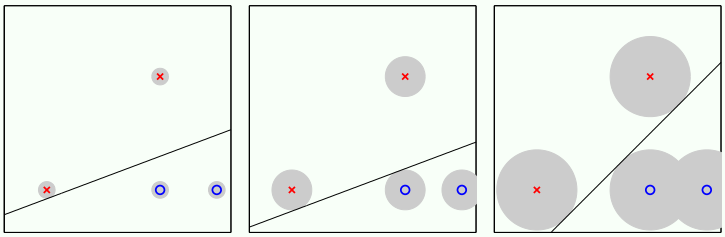
Two questions:

1. Why is bigger margin better?
2. Which $\mathbf{w}$ maximizes the margin?

Which Separator Do You Pick?

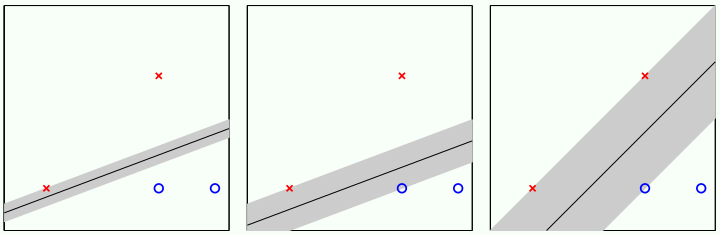


Robustness to Noisy Data



Being robust to noise (measurement error) is good (remember regularization).

Thicker Cushion Means More Robustness



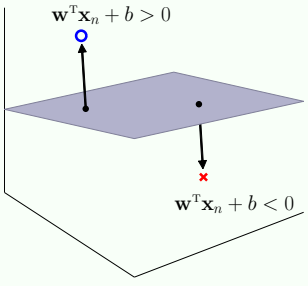
We call such hyperplanes **fat**

Two Crucial Questions

1. Can we efficiently find the fattest separating hyperplane?

2. Is a fatter hyperplane better than a thin one?

Separating The Data



Hyperplane $h = (b, \mathbf{w})$

h separates the data means:

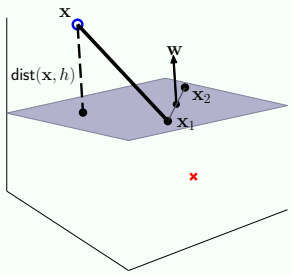
$$y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) > 0$$

By rescaling the weights and bias,

$$\min_{n=1,\ldots,N} \; y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) = 1$$

(renormalize the weights so that the signal $\mathbf{w}^{\mathrm{T}}\mathbf{x}+b$ is meaningful)

Distance to the Hyperplane



$\mathbf{w}$ is normal to the hyperplane:

$$\mathbf{w}^{\mathrm{T}}(\mathbf{x}_2 - \mathbf{x}_1) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_2 - \mathbf{w}^{\mathrm{T}}\mathbf{x}_1 = -b + b = 0.$$

(because $\mathbf{w}^{\mathrm{T}}\mathbf{x} = -b$ on the hyperplane)

Unit normal $\mathbf{u} = \mathbf{w}/\|\mathbf{w}\|$.

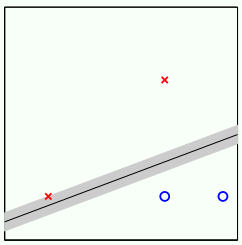
$$\begin{aligned}\mathsf{dist}(\mathbf{x}, h) &= |\mathbf{u}^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_1)| \\ &= \frac{1}{\|\mathbf{w}\|} \cdot |\mathbf{w}^{\mathrm{T}}\mathbf{x} - \mathbf{w}^{\mathrm{T}}\mathbf{x}_1| \\ &= \frac{1}{\|\mathbf{w}\|} \cdot |\mathbf{w}^{\mathrm{T}}\mathbf{x} + b|\end{aligned}$$

x_1 is on the plane so $w^T x_1 + b = 0$

Fatness of a Separating Hyperplane

$$\mathsf{dist}(\mathbf{x}, h) = \frac{1}{\|\mathbf{w}\|} \cdot |\mathbf{w}^{\mathrm{T}}\mathbf{x} + b|$$

Fatness = Distance to the closest point



Since $|\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b| = |y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b)| = y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b)$,

$$\mathsf{dist}(\mathbf{x}_n, h) = \frac{1}{\|\mathbf{w}\|} \cdot y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b).$$

$$\begin{aligned}\text{Fatness} &= \min_n \mathsf{dist}(\mathbf{x}_n, h) \\ &= \frac{1}{\|\mathbf{w}\|} \cdot \min_n y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) && \longleftarrow \text{separation condition} \\ &= \frac{1}{\|\mathbf{w}\|} && \longleftarrow \text{the } \textit{margin } \gamma(h)\end{aligned}$$

The optimization problem

Maximize $\frac{1}{\|\mathbf{w}\|}$

subject to $\min_{n=1,2,\ldots,N} |\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = 1$

Notice: $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b)$

Minimize $\frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$

subject to $y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1$ for $n = 1, 2, \ldots, N$

SVM

(Solution - QP)

Example – Our Toy Data Set

$$\mathrm{X} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

$y_n(\mathbf{w}^\mathrm{T}\mathbf{x}_n + b) \geq 1$

$$\begin{aligned} -b &\geq 1 \quad (i) \\ -(2w_1 + 2w_2 + b) &\geq 1 \quad (ii) \\ 2w_1 + b &\geq 1 \quad (iii) \\ 3w_1 + b &\geq 1 \quad (iv) \end{aligned}$$

(i) and (iii) gives $w_1 \geq 1$

(ii) and (iii) gives $w_2 \leq -1$

So, $\frac{1}{2}(w_1^2 + w_2^2) \geq 1$ $(b = -1, w_1 = 1, w_2 = -1)$

Optimal Hyperplane

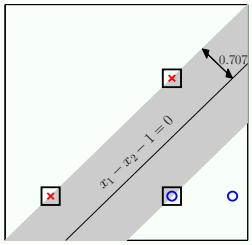
$$g(\mathbf{x}) = \mathrm{sign}(x_1 - x_2 - 1)$$

margin: $\frac{1}{\|\mathbf{w}^*\|} = \frac{1}{\sqrt{2}} \approx 0.707.$

For data points (i), (ii) and (iii) $y_n(\mathbf{w}^{*\mathrm{T}}\mathbf{x}_n + b^*) = 1$

↑

Support Vectors



Maximum Margin Hyperplane is QP

$$\underset{b,\mathbf{w}}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\mathrm{T}\mathbf{w}$$

subject to: $y_n(\mathbf{w}^\mathrm{T}\mathbf{x}_n + b) \geq 1$ for $n = 1, \ldots, N$.

$$\underset{\mathbf{u}\in\mathbb{R}^q}{\text{minimize}} \quad \frac{1}{2}\mathbf{u}^\mathrm{T}\mathrm{Q}\mathbf{u} + \mathbf{c}^\mathrm{T}\mathbf{u}$$

subject to: $\mathrm{A}\mathbf{u} \geq \mathbf{a}$

Quadratic Programming

$$\underset{\mathbf{u}\in\mathbb{R}^q}{\text{minimize}} \quad \frac{1}{2}\mathbf{u}^\mathrm{T}\mathrm{Q}\mathbf{u} + \mathbf{p}^\mathrm{T}\mathbf{u}$$

subject to: $\mathrm{A}\mathbf{u} \geq \mathbf{c}$

$$\mathbf{u}^* \leftarrow \mathrm{QP}(\mathrm{Q}, \mathbf{p}, \mathrm{A}, \mathbf{c})$$

$$\mathrm{X} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

$y_n(\mathbf{w}^\mathrm{T}\mathbf{x}_n + b) \geq 1$

$$\begin{aligned} -b &\geq 1 \quad (i) \\ -(2w_1 + 2w_2 + b) &\geq 1 \quad (ii) \\ 2w_1 + b &\geq 1 \quad (iii) \\ 3w_1 + b &\geq 1 \quad (iv) \end{aligned}$$

$$\mathrm{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathrm{A} = \begin{bmatrix} -1 & 0 & 0 \\ -1 & -2 & -2 \\ 1 & 2 & 0 \\ 1 & 3 & 0 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Use your QP-solver to give

$(b^*, w_1^*, w_2^*) = (-1, 1, -1)$

Constrained optimization

Minimize $\frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$

subject to $y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1$ for $n = 1, 2, \ldots, N$

$\mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R}$

Lagrange? inequality constraints $\Longrightarrow$ KKT

What is New? Not quite the traditional optimization problem with equality constraints. The constraints here are inequality. KKT to the rescue.

Lagrange relaxation and KKT conditions

We consider the optimization problem

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{s.t.} && g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m \end{aligned}$$

where $f(\mathbf{x})$ and $g_i(\mathbf{x})$ are real valued functions.

If $\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) & \ldots & g_m(\mathbf{x}) \end{bmatrix}^{\mathsf{T}}$ then (1) can be written

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{s.t.} && \mathbf{g}(\mathbf{x}) \leq \mathbf{0}. \end{aligned}$$

The idéa behind Lagrange relaxation is to put non-negative prices $y_i \geq 0$, on the constraints and then add these to the objective function.

This gives the (unconstrained) optimization problem:

$$\text{minimize} \quad f(\mathbf{x}) + \sum_{i=1}^{m} y_i g_i(\mathbf{x}) \qquad (3)$$

which using $\mathbf{y} = \begin{bmatrix} y_1 & \ldots & y_m \end{bmatrix}^{\mathsf{T}}$ can be written

$$\text{minimize} \quad f(\mathbf{x}) + \mathbf{y}^{\mathsf{T}}\mathbf{g}(\mathbf{x}) \qquad \boxed{L(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + \mathbf{y}^{\mathsf{T}}\mathbf{g}(\mathbf{x})}$$

Theorem 1 (Weak duality). *For an arbitrary* $\mathbf{y} \geq \mathbf{0}$ *it holds that* $\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}) \leq f(\hat{\mathbf{x}})$, *where* $\hat{\mathbf{x}}$ *is an optimal solution to (2).*

Proof: Since $\hat{\mathbf{x}}$ is a feasible solution to (2) it holds that $\mathbf{g}(\hat{\mathbf{x}}) \leq \mathbf{0}$. We get

$$\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}) \leq L(\hat{\mathbf{x}}, \mathbf{y}) = f(\hat{\mathbf{x}}) + \underbrace{\mathbf{y}^{\mathsf{T}}}_{\geq 0} \underbrace{\mathbf{g}(\hat{\mathbf{x}})}_{\leq 0} \leq f(\hat{\mathbf{x}}). \tag{4}$$

- minimizing the Lagrange function provides lower bounds to the optimization problem (2).
- By an appropriate choice of $\mathbf{y}$ a good approximation of the optimal solution to (2) is searched for. In practical algorithms one tries to solve $\max_{\mathbf{y} \geq \mathbf{0}} \min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y})$. The next theorem gives conditions for the Lagrange multiplicator providing equality in (4).

Global optimality conditions

Theorem 2. *If* $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \mathbf{R}^n \times \mathbf{R}^m$ *satisfies the conditions*

(1) $L(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \min_{x} L(\mathbf{x}, \hat{\mathbf{y}})$,

(2) $\mathbf{g}(\hat{\mathbf{x}}) \leq \mathbf{0}$,

(3) $\hat{\mathbf{y}} \geq 0$,

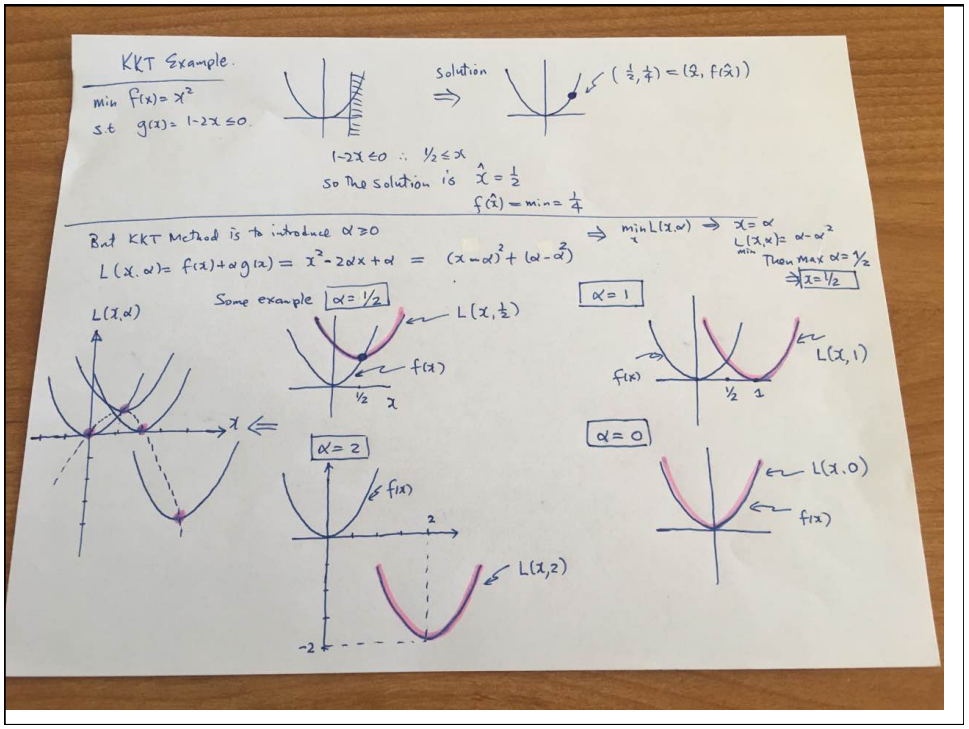
(4) $\hat{\mathbf{y}}^{\mathsf{T}} \mathbf{g}(\hat{\mathbf{x}}) = 0$.

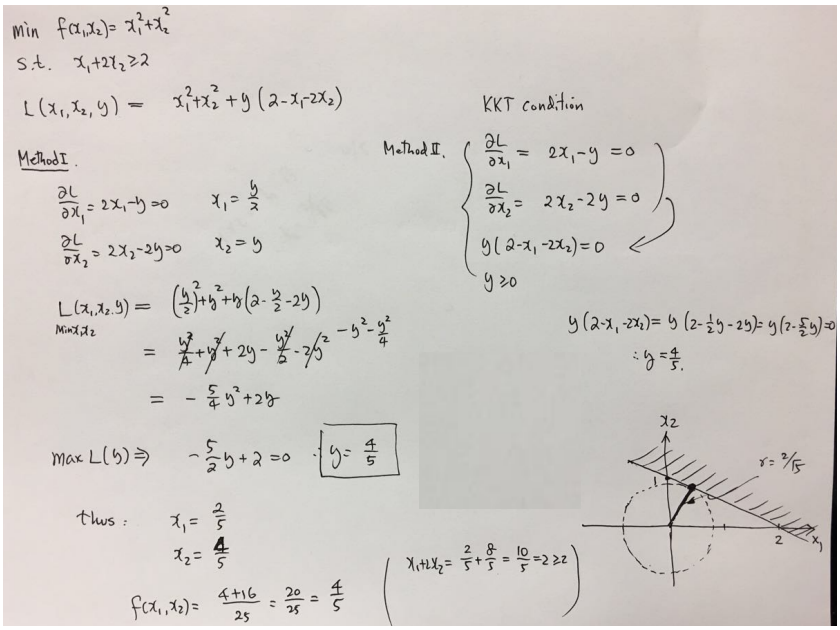
then $\hat{\mathbf{x}}$ *is an optimal solution to (2).*

Proof: *If* $\mathbf{x}$ *is an arbitrary feasible solution to (2) it holds that* $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, *which shows that*

$$f(\mathbf{x}) \geq f(\mathbf{x}) + \hat{\mathbf{y}}^{\mathsf{T}} \mathbf{g}(\mathbf{x}) = L(\mathbf{x}, \hat{\mathbf{y}}) \geq L(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = f(\hat{\mathbf{x}})$$

where the first inequality follows from (3) *and* $g(\mathbf{x}) \leq \mathbf{0}$, *the second inequality follows from* (1), *and the last one from* (4).


KKT Example.
min $f(x)=x^2$
s.t $g(x)=1-2x\le 0$
Solution
$(\frac{1}{2},\frac{1}{4})=(\hat{x},f(\hat{x}))$
$1-2x\le 0 \therefore \frac{1}{2}\le x$
So the solution is $\hat{x}=\frac{1}{2}$
$f(\hat{x})=\min=\frac{1}{4}$
But KKT Method is to introduce $\alpha\ge 0$
$L(x,\alpha)=f(x)+\alpha g(x)=x^2-2\alpha x+\alpha=(x-\alpha)^2+(\alpha-\alpha^2)$
$\Rightarrow \min_x L(x,\alpha) \rightarrow x=\alpha$
$L_{\min}(x,\alpha)=\alpha-\alpha^2$
Then max $\alpha=\frac{1}{2}$
$\Rightarrow x=\frac{1}{2}$
Some example
$\alpha=\frac{1}{2}$
$\alpha=1$
$\alpha=2$
$\alpha=0$
$L(x,\alpha)$
$L(x,\frac{1}{2})$
$L(x,1)$
$L(x,0)$
$L(x,2)$
f(x)
x


min $f(x_1,x_2)=x_1^2+x_2^2$
s.t. $x_1+2x_2\ge 2$
$L(x_1,x_2,y)=x_1^2+x_2^2+y(2-x_1-2x_2)$
KKT condition
Method I.
$\frac{\partial L}{\partial x_1}=2x_1-y=0 \quad x_1=\frac{y}{2}$
$\frac{\partial L}{\partial x_2}=2x_2-2y=0 \quad x_2=y$
$L(x_1,x_2,y)=\left(\frac{y}{2}\right)^2+y^2+y\left(2-\frac{y}{2}-2y\right)$
$\min_{x_1,x_2}$
$=\frac{y^2}{4}+y^2+2y-\frac{y^2}{2}-2y^2-\frac{y^2}{4}$
$=-\frac{5}{4}y^2+2y$
$\max L(y) \Rightarrow -\frac{5}{2}y+2=0 \therefore y=\frac{4}{5}$
thus: $x_1=\frac{2}{5}$
$x_2=\frac{4}{5}$
$f(x_1,x_2)=\frac{4+16}{25}=\frac{20}{25}=\frac{4}{5}$
Method II.
$\frac{\partial L}{\partial x_1}=2x_1-y=0$
$\frac{\partial L}{\partial x_2}=2x_2-2y=0$
$y(2-x_1-2x_2)=0$
$y\ge 0$
$y(2-x_1-2x_2)=y(2-\frac{1}{2}y-2y)=y(2-\frac{5}{2}y)=0$
$\therefore y=\frac{4}{5}$
$x_1+2x_2=\frac{2}{5}+\frac{8}{5}=\frac{10}{5}=2\ge 2$
x_2
x_1
$r=2/5$
2

Lagrange formulation

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

w.r.t. $\mathbf{w}$ and b and maximize w.r.t. each $\alpha_n \geq 0$

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = \mathbf{0}$$

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

Substituting ...

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n \qquad \text{and} \qquad \sum_{n=1}^{N} \alpha_n y_n = 0$$

in the Lagrangian $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{n=1}^{N} \alpha_n (y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) - 1)$

we get $\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$

Maximize w.r.t. to $\boldsymbol{\alpha}$ subject to $\alpha_n \geq 0$ for $n = 1, \cdots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

The solution - quadratic programming

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^{\mathsf{T}} \underbrace{\begin{bmatrix} y_1y_1\,\mathbf{x}_1^{\mathsf{T}}\mathbf{x}_1 & y_1y_2\,\mathbf{x}_1^{\mathsf{T}}\mathbf{x}_2 & \dots & y_1y_N\,\mathbf{x}_1^{\mathsf{T}}\mathbf{x}_N \\ y_2y_1\,\mathbf{x}_2^{\mathsf{T}}\mathbf{x}_1 & y_2y_2\,\mathbf{x}_2^{\mathsf{T}}\mathbf{x}_2 & \dots & y_2y_N\,\mathbf{x}_2^{\mathsf{T}}\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_Ny_1\,\mathbf{x}_N^{\mathsf{T}}\mathbf{x}_1 & y_Ny_2\,\mathbf{x}_N^{\mathsf{T}}\mathbf{x}_2 & \dots & y_Ny_N\,\mathbf{x}_N^{\mathsf{T}}\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}} \boldsymbol{\alpha} + \underbrace{(-\mathbf{1}^{\mathsf{T}})}_{\text{linear}}\boldsymbol{\alpha}$$

subject to $\underbrace{\mathbf{y}^{\mathsf{T}}\boldsymbol{\alpha} = 0}_{\text{linear constraint}}$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \leq \boldsymbol{\alpha} \leq \underbrace{\infty}_{\text{upper bounds}}$$

Still has to call a QP solver, but the problem is must simplified with a simple constraint.

QP hands us $\boldsymbol{\alpha}$

Solution: $\boldsymbol{\alpha} = \alpha_1, \cdots, \alpha_N$

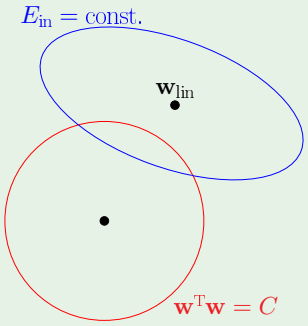
$$\Longrightarrow \quad \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

KKT condition: For $n = 1, \cdots, N$

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

We saw this before!

$\alpha_n > 0 \implies \mathbf{x}_n$ is a **support vector**



Support vectors

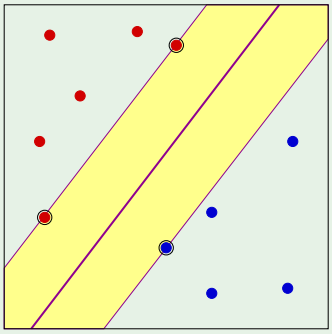
Closest $\mathbf{x}_n$'s to the plane: achieve the margin

$$\implies \quad y_n \left(\mathbf{w}^{\mathsf{T}} \mathbf{x}_n + b\right) = 1$$

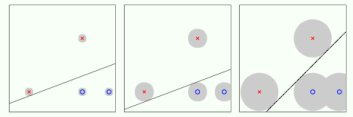
$$\mathbf{w} \;=\; \sum_{\mathbf{x}_n \text{ is SV}} \alpha_n y_n \mathbf{x}_n$$

Solve for b using any SV:

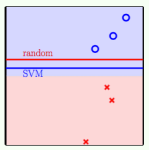
$$y_n \left(\mathbf{w}^{\mathsf{T}} \mathbf{x}_n + b\right) = 1$$

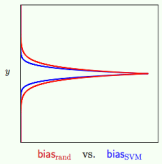


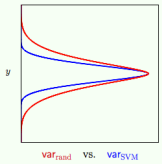
Evidence that Larger Margin is Better

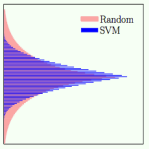


Bias and Variance









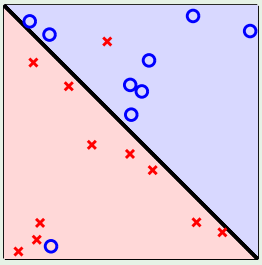
	Random	SVM	
bias	0.02	0.015	**-0.005**
var	0.059	0.038	**-0.021**
E_{out}	0.079	0.053	**-0.026**

SVM

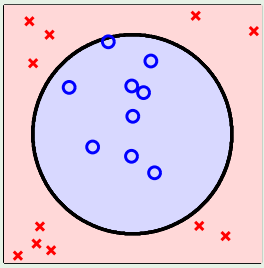
(Extensions)

Two types of non-separable

slightly:



seriously:



SVM

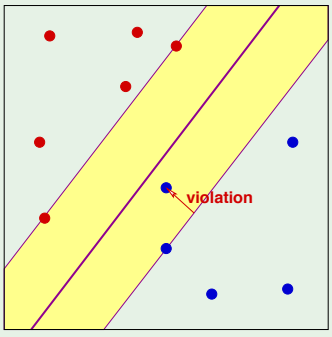
(Soft)

Error measure

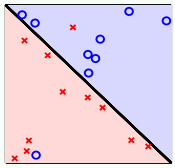
Margin violation: $y_n \left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1$ fails

Quantify: $y_n \left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1 \; - \; \xi_n \qquad \xi_n \geq 0$

Total violation $= \sum_{n=1}^{N} \xi_n$



Soft Margin SVM



tolerate error

'soft in-sample error'

$$\underset{b,\mathbf{w},\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$$

'soft' error on $(\mathbf{x}_n, y_n)$

$$\text{subject to:} \quad y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 - \xi_n$$

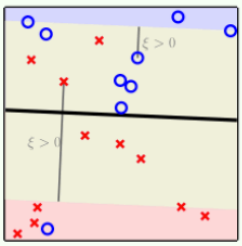
$$\xi_n \geq 0 \qquad \text{for } n = 1, \ldots, N$$

Trades off 'soft in-sample error' $\sum_{n=1}^{N}\xi_n$ with weight norm $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ ← regularization

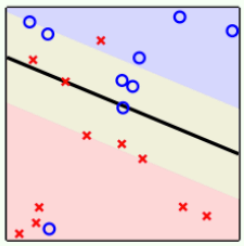
C plays the role of a regularization parameter ($\lambda \sim \frac{1}{C}$)

Choice of C is important - similar to choice of λ in regularization

Non-Separable Data



$C = 1$



$C = 500$

$$\underset{b,\mathbf{w},\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$$

$$\text{subject to:} \quad y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 - \xi_n$$

$$\xi_n \geq 0 \qquad \text{for } n = 1, \ldots, N$$

The new optimization

$$\text{Minimize} \quad \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$$

$$\text{subject to} \quad y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1 - \xi_n \quad \text{for} \quad n = 1, \ldots, N$$

$$\text{and} \quad \xi_n \geq 0 \quad \text{for} \quad n = 1, \ldots, N$$

$$\mathbf{w} \in \mathbb{R}^d \quad , \quad b \in \mathbb{R} \quad , \quad \boldsymbol{\xi} \in \mathbb{R}^N$$

Lagrange formulation

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left(y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) - 1 + \xi_n\right) - \sum_{n=1}^{N}\beta_n\,\xi_n$$

Minimize w.r.t. $\mathbf{w}$, b, and $\boldsymbol{\xi}$ and maximize w.r.t. each $\alpha_n \geq 0$ and $\beta_n \geq 0$

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{n=1}^{N}\alpha_n y_n \mathbf{x}_n = \mathbf{0}$$

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

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = C - \alpha_n - \beta_n = 0$$

and the solution is ...

Maximize $\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$ w.r.t. to $\boldsymbol{\alpha}$

subject to $0 \leq \alpha_n \leq C$ for $n = 1, \cdots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

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

minimizes $\frac{1}{2} \mathbf{w}^{\mathsf{T}} \mathbf{w} + C \sum_{n=1}^{N} \xi_n$

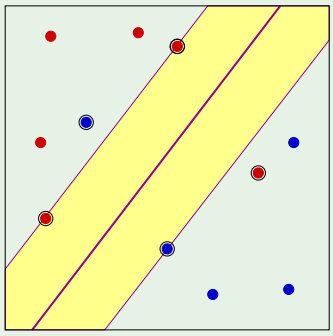
Types of support vectors

margin support vectors $(0 < \alpha_n < C)$

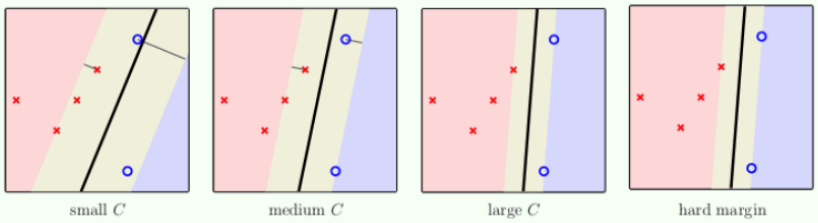
$y_n (\mathbf{w}^{\mathsf{T}} \mathbf{x}_n + b) = 1 \qquad (\xi_n = 0)$

non-margin support vectors $(\alpha_n = C)$

$y_n (\mathbf{w}^{\mathsf{T}} \mathbf{x}_n + b) < 1 \qquad (\xi_n > 0)$



Soft Margin SVM With Separable Data



$$\underset{b,\mathbf{w},\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$$

$$\text{subject to:} \quad y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 - \xi_n$$

$$\xi_n \geq 0 \qquad \text{for } n = 1, \ldots, N$$

Choice of C is **IMPORTANT**

Two technical observations

1. Hard margin: What if data is not linearly separable?

   "primal $\longrightarrow$ dual" breaks down

2. $\mathcal{Z}$: What if there is w_0?

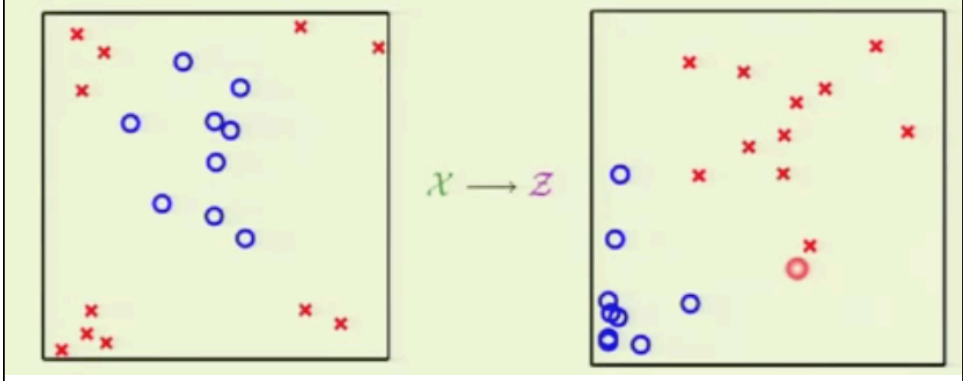
   All goes to b and $w_0 \rightarrow 0$

SVM

(Extension: Kernel)

z instead of x

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{z}_n^{\mathsf{T}} \mathbf{z}_m$$



What do we need from the $\mathcal{Z}$ space?

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{z}_n^{\mathsf{T}} \mathbf{z}_m$$

Constraints: $\alpha_n \geq 0$ for $n = 1, \cdots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

$$\boxed{g(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^{\mathsf{T}} \mathbf{z} + b\right)} \qquad \text{need } \mathbf{z}_n^{\mathsf{T}} \mathbf{z}$$

where $\mathbf{w} = \sum_{\mathbf{z}_n \text{ is SV}} \alpha_n y_n \mathbf{z}_n$

and b: $y_m \left(\mathbf{w}^{\mathsf{T}} \mathbf{z}_m + b\right) = 1$ need $\mathbf{z}_n^{\mathsf{T}} \mathbf{z}_m$

Generalized inner product

Given two points $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}$, we need $\mathbf{z}^{\mathsf{T}} \mathbf{z}'$

Let $\mathbf{z}^{\mathsf{T}} \mathbf{z}' = K(\mathbf{x}, \mathbf{x}')$ (the kernel) "inner product" of $\mathbf{x}$ and $\mathbf{x}'$

Example: $\mathbf{x} = (x_1, x_2) \longrightarrow$ 2nd-order Φ

$$\mathbf{z} = \Phi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{z}^{\mathsf{T}} \mathbf{z}' = 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x'_1 x_2 x'_2$$

The trick

Can we compute $K(\mathbf{x}, \mathbf{x}')$ without transforming $\mathbf{x}$ and $\mathbf{x}'$?

Example: Consider $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^{\mathrm{T}}\mathbf{x}')^2 = (1 + x_1x'_1 + x_2x'_2)^2$

$$= 1 + x_1^2x'^2_1 + x_2^2x'^2_2 + 2x_1x'_1 + 2x_2x'_2 + 2x_1x'_1x_2x'_2$$

This is an inner product!

$$(\ 1\ ,\ x_1^2\ ,\ x_2^2\ ,\ \sqrt{2}\,x_1\ ,\ \sqrt{2}\,x_2\ ,\ \sqrt{2}\,x_1x_2\ )$$

$$(\ 1\ ,\ x'^2_1\ ,\ x'^2_2\ ,\ \sqrt{2}x'_1\ ,\ \sqrt{2}x'_2\ ,\ \sqrt{2}x'_1x'_2\ )$$

The Kernel $K(\cdot, \cdot)$ for a Transform $\Phi(\cdot)$

The Kernel tells you how to compute the inner product in $\mathcal{Z}$-space

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^{\mathrm{T}}\Phi(\mathbf{x}') = \mathbf{z}^{\mathrm{T}}\mathbf{z}'$$

Example: 2nd-order polynomial transform

$$\Phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^{\mathrm{T}}\Phi(\mathbf{x}') = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} x'_1 \\ x'_2 \\ x'^2_1 \\ \sqrt{2}x'_1x'_2 \\ x'^2_2 \end{bmatrix} \quad \leftarrow O(d^2)$$

$$= x_1x'_1 + x_2x'_2 + x_1^2x'^2_1 + 2x_1x_2x'_1x'_2 + x_2^2x'^2_2$$

$$= \left(\frac{1}{2} + \mathbf{x}^{\mathrm{T}}\mathbf{x}'\right)^2 - \frac{1}{4}$$

↑ computed quickly in $\mathcal{X}$-space, in $O(d)$

Gaussian kernel →

The polynomial kernel

$\mathcal{X} = \mathbb{R}^d$ and $\Phi : \mathcal{X} \to \mathcal{Z}$ is polynomial of order Q

The "equivalent" kernel $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\mathsf{T}\mathbf{x}')^Q$

$$= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^Q$$

Compare for $d = 10$ and $Q = 100$

Can adjust scale: $K(\mathbf{x}, \mathbf{x}') = (a\mathbf{x}^\mathsf{T}\mathbf{x}' + b)^Q$

We only need $\mathcal{Z}$ to exist!

If $K(\mathbf{x}, \mathbf{x}')$ is an inner product in some space $\mathcal{Z}$, we are good.

Example: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$

Infinite-dimensional $\mathcal{Z}$: take simple case

$$K(x, x') = \exp\left(-(x - x')^2\right)$$

$$= \exp\left(-x^2\right) \exp\left(-x'^2\right) \underbrace{\sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}}_{\exp(2xx')}$$

The Gaussian Kernel is Infinite-Dimensional

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x}-\mathbf{x}'\|^2}$$

Example: Gaussian Kernel in 1-dimension

$$\Phi(\mathbf{x}) = \begin{bmatrix} e^{-x^2}\sqrt{\frac{2^0}{0!}} \\ e^{-x^2}\sqrt{\frac{2^1}{1!}}x \\ e^{-x^2}\sqrt{\frac{2^2}{2!}}x^2 \\ e^{-x^2}\sqrt{\frac{2^3}{3!}}x^3 \\ e^{-x^2}\sqrt{\frac{2^4}{4!}}x^4 \\ \vdots \end{bmatrix}$$

(infinite dimensional Φ)

$$K(x, x') = \Phi(x)^{\mathrm{T}}\Phi(x') = \begin{bmatrix} e^{-x^2}\sqrt{\frac{2^0}{0!}} \\ e^{-x^2}\sqrt{\frac{2^1}{1!}}x \\ e^{-x^2}\sqrt{\frac{2^2}{2!}}x^2 \\ e^{-x^2}\sqrt{\frac{2^3}{3!}}x^3 \\ e^{-x^2}\sqrt{\frac{2^4}{4!}}x^4 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} e^{-x'^2}\sqrt{\frac{2^0}{0!}} \\ e^{-x'^2}\sqrt{\frac{2^1}{1!}}x' \\ e^{-x'^2}\sqrt{\frac{2^2}{2!}}x'^2 \\ e^{-x'^2}\sqrt{\frac{2^3}{3!}}x'^3 \\ e^{-x'^2}\sqrt{\frac{2^4}{4!}}x'^4 \\ \vdots \end{bmatrix}$$

$$= e^{-x^2}e^{-x'^2}\sum_{i=0}^{\infty}\frac{(2xx')^i}{i!}$$

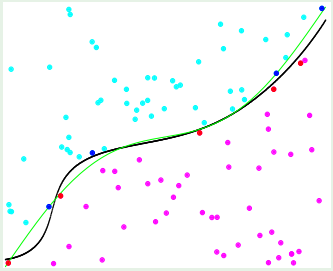
$$= e^{-(x-x')^2}$$

This kernel in action

Slightly non-separable case:

Transforming $\mathcal{X}$ into ∞-dimensional $\mathcal{Z}$

Overkill? Count the support vectors



Kernel formulation of SVM

Remember quadratic programming? The only difference now is:

$$\underbrace{\begin{bmatrix} y_1y_1\,K(\mathbf{x}_1,\mathbf{x}_1) & y_1y_2\,K(\mathbf{x}_1,\mathbf{x}_2) & \dots & y_1y_N K(\mathbf{x}_1,\mathbf{x}_N) \\ y_2y_1\,K(\mathbf{x}_2,\mathbf{x}_1) & y_2y_2\,K(\mathbf{x}_2,\mathbf{x}_2) & \dots & y_2y_N K(\mathbf{x}_2,\mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ y_Ny_1 K(\mathbf{x}_N,\mathbf{x}_1) & y_Ny_2 K(\mathbf{x}_N,\mathbf{x}_2) & \dots & y_Ny_N K(\mathbf{x}_N,\mathbf{x}_N) \end{bmatrix}}_{\text{quadratic coefficients}}$$

Everything else is the same.

The final hypothesis

Express $g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{\mathsf{T}}\mathbf{z} + b\right)$ in terms of $K(-,-)$

$$\mathbf{w} = \sum_{\mathbf{z}_n \text{ is SV}} \alpha_n y_n \mathbf{z}_n \quad \Longrightarrow \quad g(\mathbf{x}) = \text{sign}\left(\sum_{\alpha_n>0} \alpha_n y_n K(\mathbf{x}_n,\mathbf{x}) + b\right)$$

where $b = y_m - \sum_{\alpha_n>0} \alpha_n y_n K(\mathbf{x}_n,\mathbf{x}_m)$

for any support vector $(\alpha_m > 0)$

How do we know that $\mathcal{Z}$ exists ...

... for a given $K(\mathbf{x}, \mathbf{x}')$? valid kernel

Three approaches:

1. By construction
2. Math properties (*Mercer's condition*)
3. Who cares? ☺

Design your own kernel

$K(\mathbf{x}, \mathbf{x}')$ is a valid kernel iff

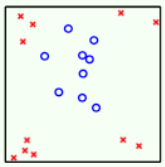
1. It is symmetric and 2. The matrix:

$$\begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ K(\mathbf{x}_N, \mathbf{x}_1) & K(\mathbf{x}_N, \mathbf{x}_2) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

is positive semi-definite

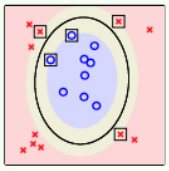
for any $\mathbf{x}_1, \cdots, \mathbf{x}_N$ (Mercer's condition)

The Kernel Allows Us to Bypass $\mathcal{Z}$-space



$\mathbf{x}_n \in \mathcal{X}$

$\downarrow K(\cdot,\cdot)$



$$g(\mathbf{x}) = \text{sign}\left(\sum_{\alpha_n^*>0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}) + b^*\right)$$

$$b^* = y_s - \sum_{\alpha_n^*>0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}_s)$$

1: **Input:** X, $\mathbf{y}$, regularization parameter C

2: Compute G: $\mathrm{G}_{nm} = y_n y_m K(\mathbf{x}_n, \mathbf{x}_m)$.

3: Solve (QP):

$$\left.\begin{array}{ll} \underset{\boldsymbol{\alpha}}{\text{minimize:}} & \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\mathrm{G}\boldsymbol{\alpha} - \mathbf{1}^{\mathrm{T}}\boldsymbol{\alpha} \\ \text{subject to:} & \mathbf{y}^{\mathrm{T}}\boldsymbol{\alpha} = 0 \\ & \mathbf{C} \geq \boldsymbol{\alpha} \geq \mathbf{0} \end{array}\right\} \longrightarrow \begin{array}{l} \boldsymbol{\alpha}^\star \\ \text{index } s : C > \alpha_s^* > 0 \end{array}$$

4: $b^* = y_s - \sum_{\alpha_n^*>0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}_s)$

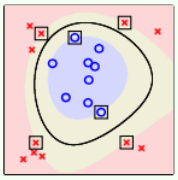
5: The final hypothesis is

$$g(\mathbf{x}) = \text{sign}\left(\sum_{\alpha_n^*>0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}) + b^*\right)$$

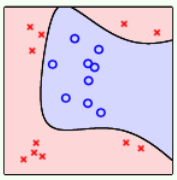
The Kernel-Support Vector Machine

Overfitting

SVM



Regression



high $\tilde{d}$ → complicated separator

small # support vectors → low effective complexity

Can go to high (infinite) $\tilde{d}$

Computation

Inner products with Kernel

$K(\cdot,\cdot)$

high $\tilde{d}$ → expensive or infeasible computation

kernel → computationally feasible to go to high $\tilde{d}$

Can go to high (infinite) $\tilde{d}$

SVM

(Radial Basis Functions)

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 16: Radial Basis Functions

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Thursday, May 24, 2012

Outline

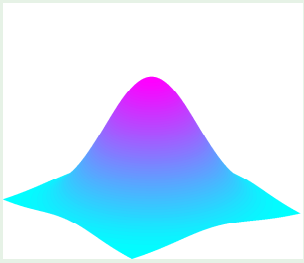
- RBF and nearest neighbors
- RBF and neural networks
- RBF and kernel methods
- RBF and regularization

Basic RBF model

Each $(\mathbf{x}_n, y_n) \in \mathcal{D}$ influences $h(\mathbf{x})$ based on $\underbrace{\|\mathbf{x} - \mathbf{x}_n\|}_{\text{radial}}$

Standard form:

$$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \underbrace{\exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_n\|^2\right)}_{\text{basis function}}$$



The learning algorithm

Finding $w_1, \cdots, w_N$: $\quad h(\mathbf{x}) = \sum_{n=1}^{N} w_n \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_n\|^2\right)$

based on $\mathcal{D} = (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

$E_{\text{in}} = 0$: $\quad h(\mathbf{x}_n) = y_n$ for $n = 1, \cdots, N$:

$$\sum_{m=1}^{N} w_m \exp\left(-\gamma \|\mathbf{x}_n - \mathbf{x}_m\|^2\right) = y_n$$

The solution

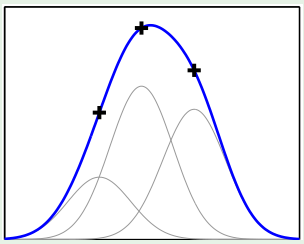
$$\sum_{m=1}^{N} w_m \exp\left(-\gamma \|\mathbf{x}_n - \mathbf{x}_m\|^2\right) = y_n \qquad N \text{ equations in } N \text{ unknowns}$$

$$\underbrace{\begin{bmatrix} \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_N\|^2) \\ \exp(-\gamma \|\mathbf{x}_2 - \mathbf{x}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_2 - \mathbf{x}_N\|^2) \\ \vdots & \vdots & \vdots \\ \exp(-\gamma \|\mathbf{x}_N - \mathbf{x}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_N - \mathbf{x}_N\|^2) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}}$$

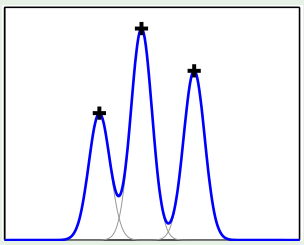
If Φ is invertible, $\boxed{\mathbf{w} = \Phi^{-1}\mathbf{y}}$ "exact interpolation"

The effect of γ

$$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_n\|^2\right)$$



small γ



large γ

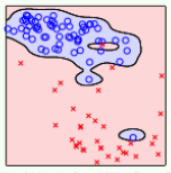
RBF-Kernel

One dimensional RBF-Kernel

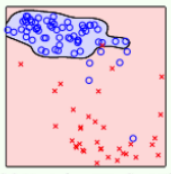
$$\Phi(x) = e^{-x^2}\begin{bmatrix} 1 \\ \sqrt{\frac{2^1}{1!}}x \\ \sqrt{\frac{2^2}{2!}}x^2 \\ \sqrt{\frac{2^3}{3!}}x^3 \\ \sqrt{\frac{2^4}{4!}}x^4 \\ \vdots \end{bmatrix}$$

$$\begin{aligned} K(x, x') &= \Phi(x)^T \Phi(x') \\ &= e^{-x^2} e^{-x'^2} \sum_{i=0}^{\infty} \frac{(2xx')^i}{i!} \quad \leftarrow \text{not feasible} \\ &= e^{-x^2} e^{-x'^2} e^{2xx'} \\ &= e^{-(x-x')^2} \end{aligned}$$

↑ computed quickly in $\mathcal{X}$-space, in $O(d)$



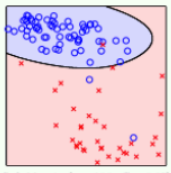
Hard Margin ($\gamma = 2000$, $C = \infty$)



Soft Margin ($\gamma = 2000$, $C = 0.25$)

d-dimensional RBF-Kernel

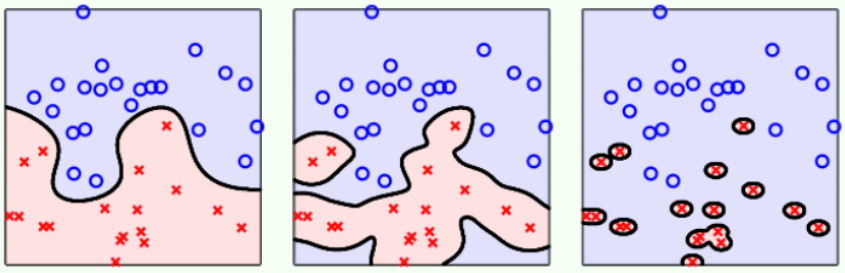
$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} \quad (\gamma > 0)$$



Soft Margin ($\gamma = 100$, $C = 0.25$)

Choosing RBF-Kernel Width γ

$$e^{-\gamma \| \mathbf{x}-\mathbf{x}' \|^2}$$



Small γ | Medium γ | Large γ!

RBF for classification

$$h(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} w_n \exp\left(-\gamma \|\mathbf{x}-\mathbf{x}_n\|^2\right)\right)$$

Learning: $\sim$ linear regression for classification

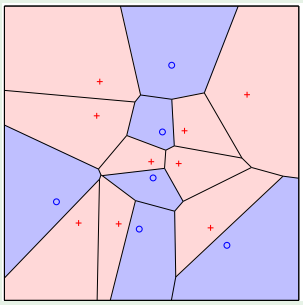
$$s = \sum_{n=1}^{N} w_n \exp\left(-\gamma \|\mathbf{x}-\mathbf{x}_n\|^2\right)$$

Minimize $(s-y)^2$ on $\mathcal{D}$ $\quad y = \pm 1$

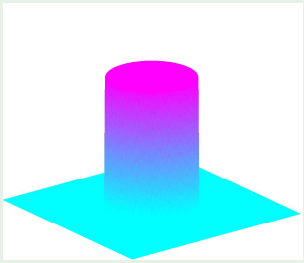
$h(\mathbf{x}) = \text{sign}(s)$

Relationship to nearest-neighbor method

Adopt the y value of a nearby point:



similar effect by a basis function:



RBF with K centers

N parameters $w_1, \cdots, w_N$ based on N data points

Use $K \ll N$ centers: $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$ instead of $\mathbf{x}_1, \cdots, \mathbf{x}_N$

$$h(\mathbf{x}) = \sum_{k=1}^{K} w_k \exp\left(-\gamma \|\mathbf{x} - \boldsymbol{\mu}_k\|^2\right)$$

1. How to choose the centers $\boldsymbol{\mu}_k$
2. How to choose the weights w_k

Choosing the centers

Minimize the distance between $\mathbf{x}_n$ and the **closest** center $\boldsymbol{\mu}_k$: K-means clustering

Split $\mathbf{x}_1, \cdots, \mathbf{x}_N$ into clusters $S_1, \cdots, S_K$

Minimize $\displaystyle\sum_{k=1}^{K} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$

Unsupervised learning ☺

NP-hard ☹

An iterative algorithm

Lloyd's algorithm: Iteratively minimize $\displaystyle\sum_{k=1}^{K} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ w.r.t. $\boldsymbol{\mu}_k, S_k$

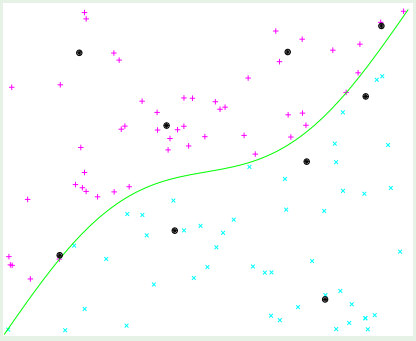
$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|S_k|} \sum_{\mathbf{x}_n \in S_k} \mathbf{x}_n$$

$$S_k \leftarrow \{\mathbf{x}_n : \|\mathbf{x}_n - \boldsymbol{\mu}_k\| \leq \text{all } \|\mathbf{x}_n - \boldsymbol{\mu}_\ell\|\}$$

Convergence $\longrightarrow$ **local minimum**

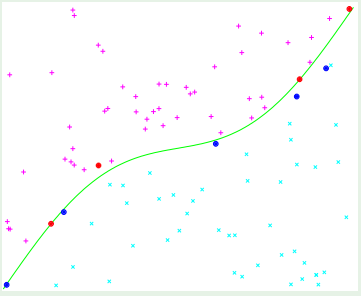
Lloyd's algorithm in action

1. Get the data points
2. Only the inputs!
3. Initialize the centers
4. Iterate
5. These are your $\boldsymbol{\mu}_k$'s

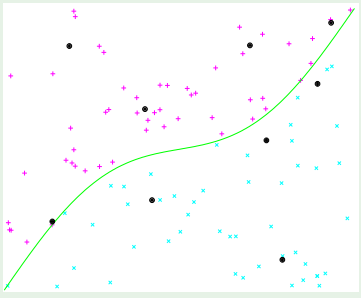


Centers versus support vectors

support vectors



RBF centers



Choosing the weights

$$\sum_{k=1}^{K} w_k \exp\left(-\gamma \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2\right) \approx y_n \qquad N \text{ equations in } K < N \text{ unknowns}$$

$$\underbrace{\begin{bmatrix} \exp(-\gamma \|\mathbf{x}_1 - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_1 - \boldsymbol{\mu}_K\|^2) \\ \exp(-\gamma \|\mathbf{x}_2 - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_2 - \boldsymbol{\mu}_K\|^2) \\ \vdots & \vdots & \vdots \\ \exp(-\gamma \|\mathbf{x}_N - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_N - \boldsymbol{\mu}_K\|^2) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}}_{\mathbf{w}} \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}}$$

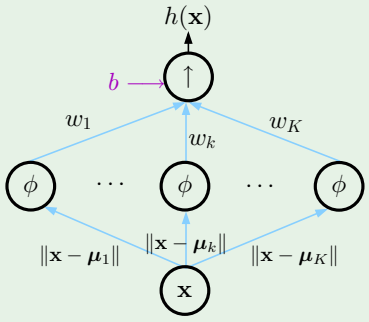
If $\Phi^{\mathrm{T}}\Phi$ is invertible, $\boxed{\mathbf{w} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathbf{y}}$ pseudo-inverse

RBF network

The "features" are $\exp\left(-\gamma \|\mathbf{x} - \boldsymbol{\mu}_k\|^2\right)$

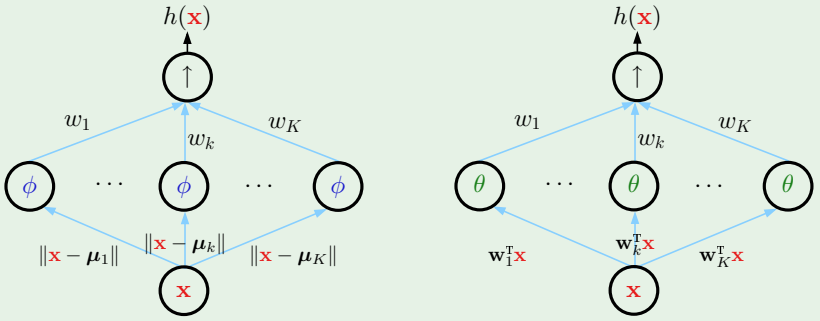
Nonlinear transform depends on $\mathcal{D}$

$\Longrightarrow$ No longer a linear model



A bias term (b or w_0) is often added

Compare to neural networks



Choosing γ

Treating γ as a parameter to be learned $\quad h(\mathbf{x}) = \sum_{k=1}^{K} w_k \exp\left(-\gamma \|\mathbf{x}-\boldsymbol{\mu}_k\|^2\right)$

Iterative approach ($\sim$ **EM algorithm** in mixture of Gaussians):

1. Fix γ, solve for $w_1, \cdots, w_K$
2. Fix $w_1, \cdots, w_K$, minimize error w.r.t. γ

We can have a different γ_k for each center $\boldsymbol{\mu}_k$