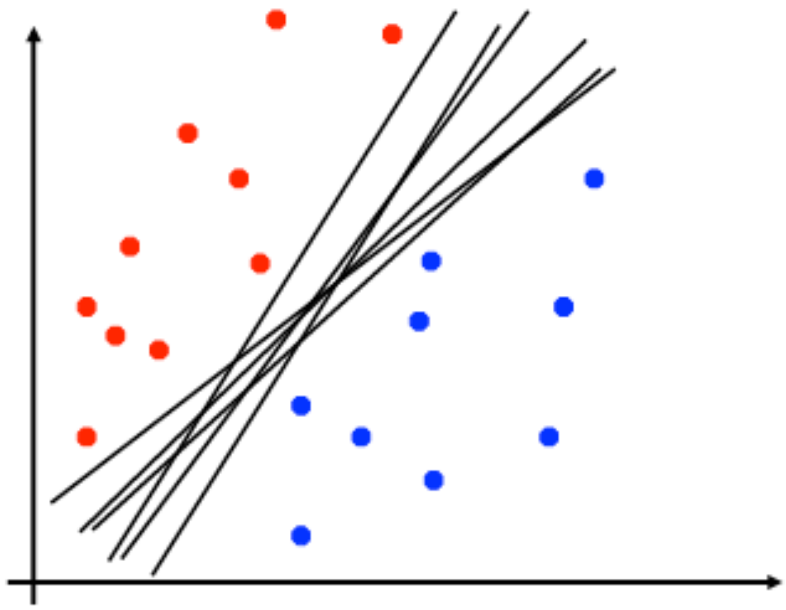
Support Vector Machines & Kernelization

Barna Saha

Most of the slides are made using David Sontag's course on machine Learning at MIT

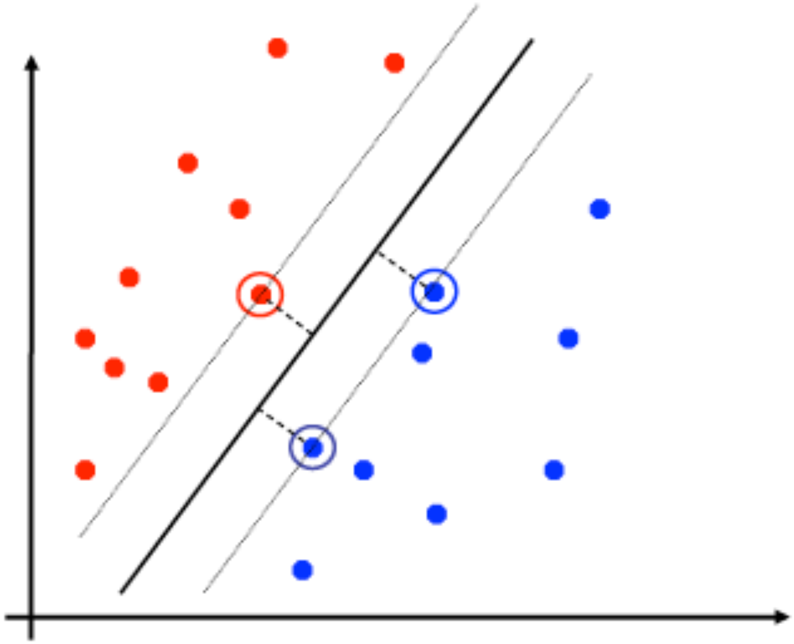
Linear Separators

- Which of these linear separators is optimal?



Support Vector Machines

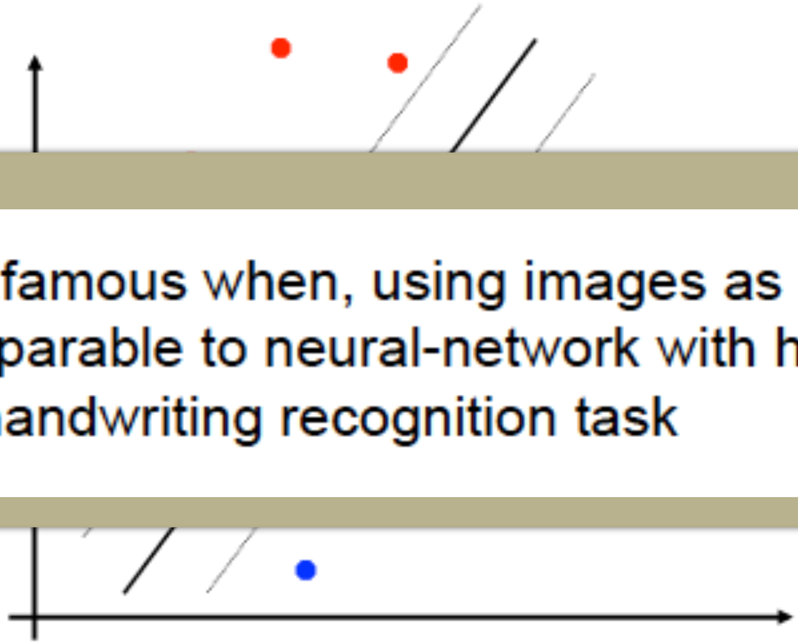
- SVMs (Vapnik, 1990's) choose the linear separator with the **largest margin**



- Good according to intuition, theory, practice

Support Vector Machines

- SVMs (Vapnik, 1990's) choose the linear separator with the **largest margin**

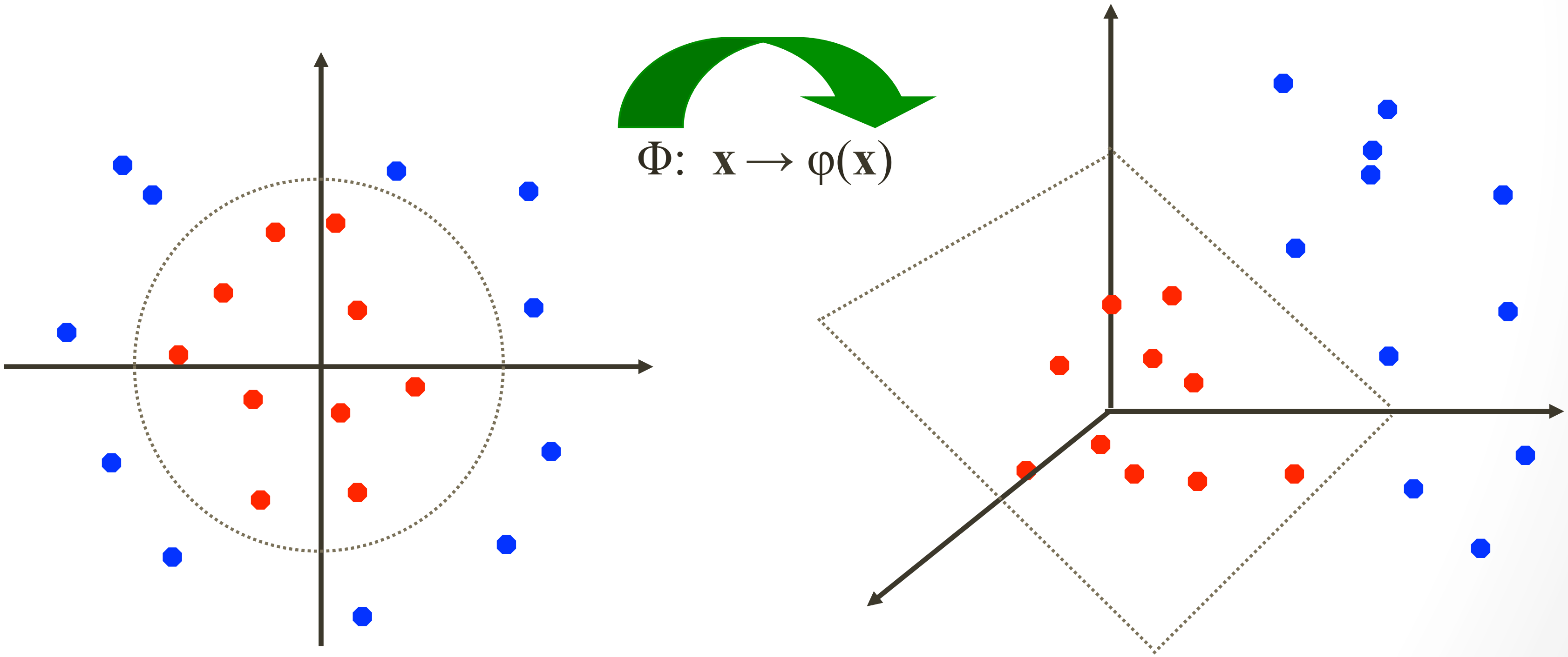


- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

- Good according to intuition, theory, practice

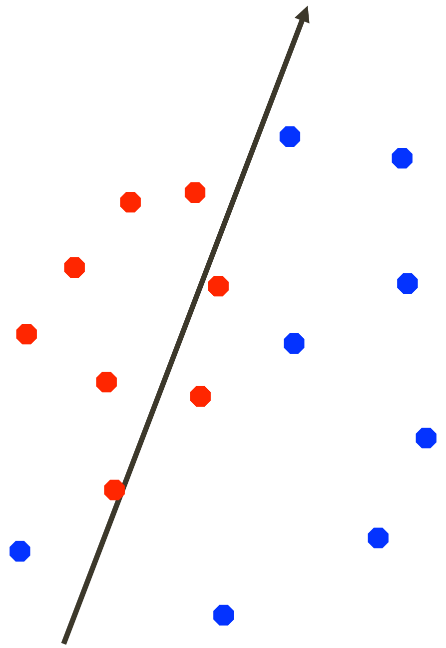
What if the data is not linearly separable?

- General idea: the original feature space can always be mapped to a different (often some higher-dimensional feature space) where the training set is separable: [x_1, x_2]→[\sqrt(x_1^2+$x_2^{2)}$, arctan(x_2/x_1)]



What if the data is not linearly separable?

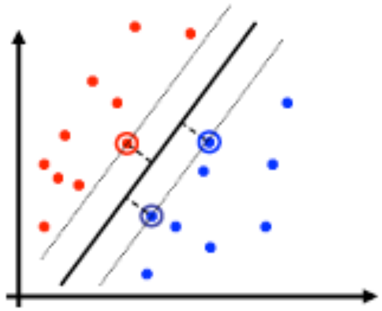
- If there is a separator which “almost” separates, find a separator that minimizes some kind of loss function.



Support vector machines: 3 key ideas

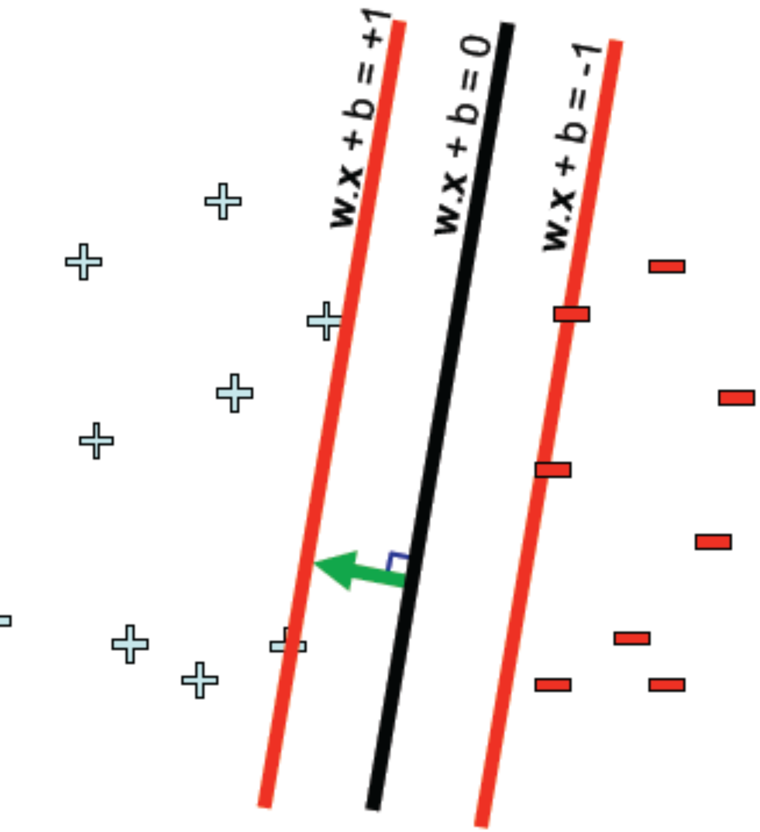
1. Use **optimization** to find solution (i.e. a hyperplane) with few errors

2. Seek **large margin** separator to improve generalization



3. Use **kernel trick** to make large feature spaces computationally efficient

Finding a perfect classifier (when one exists) using linear programming



For every data point (x_t, y_t), enforce the constraint

for $y_t = +1$, $w \cdot x_t + b \geq 1$

and for $y_t = -1$, $w \cdot x_t + b \leq -1$

Equivalently, we want to satisfy all of the linear constraints

$$y_t \left(w \cdot x_t + b \right) \geq 1 \quad \forall t$$

This *linear program* can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid

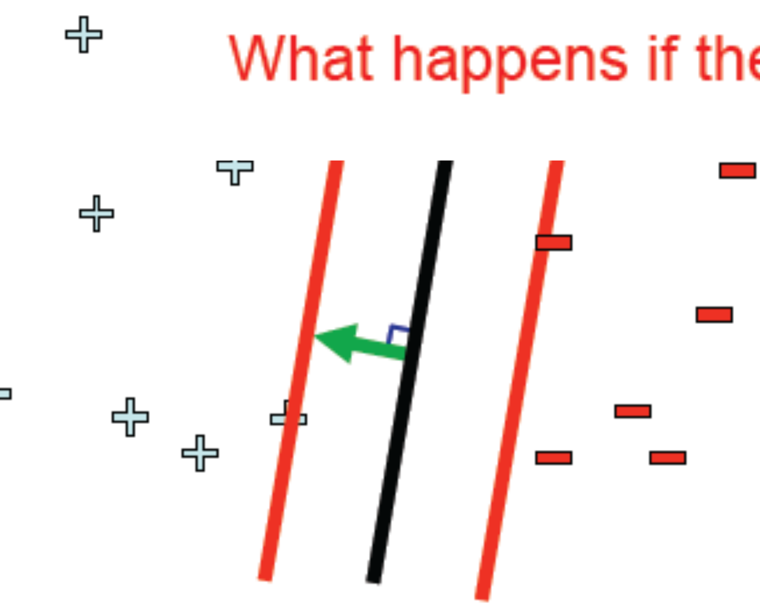
w is normal to the hyperplane w.x+b=0

Finding a perfect classifier (when one exists) using linear programming

For every data point (x_t, y_t), enforce the constraint

for $y_t = +1$, $w \cdot x_t + b \geq 1$

What happens if the data set is not linearly separable?

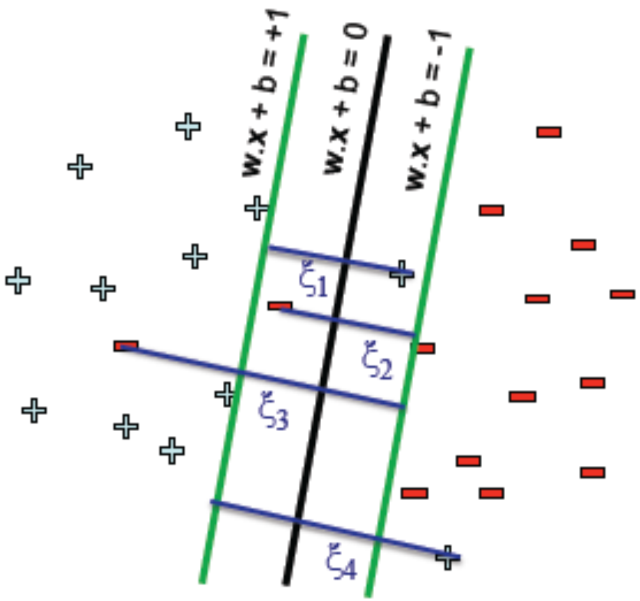


Equivalently, we want to satisfy all of the linear constraints

$$y_t\,(w \cdot x_t + b) \geq 1 \quad \forall t$$

This *linear program* can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid

Key idea #1: Allow for *slack*



$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \, \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

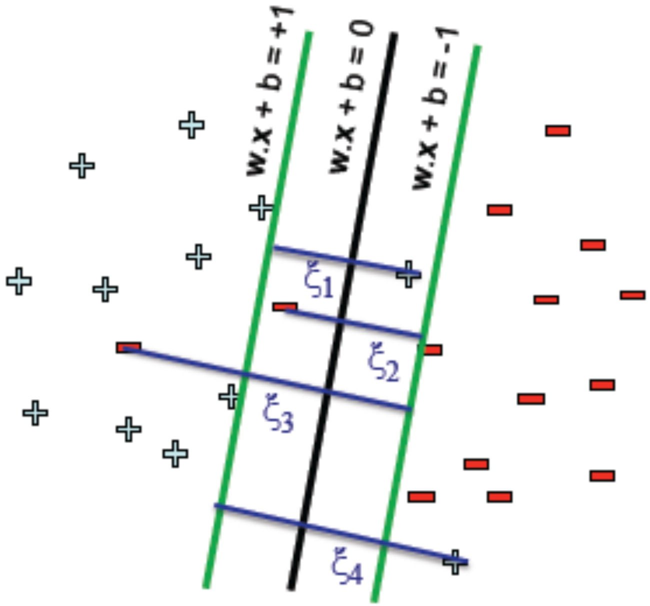
"slack variables"

We now have a linear program again, and can efficiently find its optimum

For each data point:

- If functional margin ≥ 1, don't care
- If functional margin < 1, pay linear penalty

Key idea #1: Allow for *slack*



$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \, \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

↑ "slack variables"

What is the optimal value ξ_j^* as a function of w* and b*?

If $(w \cdot x_j + b)\, y_j \geq 1$, then $\xi_j = 0$

If $(w \cdot x_j + b)\, y_j < 1$, then $\xi_j = 1 - (w \cdot x_j + b)\, y_j$

↓

$$\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$$

Sometimes written as

$$\left(1 - (w \cdot x_j + b)\, y_j\right)_+$$

Equivalent hinge loss formulation

$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \, \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

Substituting $\xi_j = \max(0, 1 - (w \cdot x_j + b)\, y_j)$ into the objective, we get:

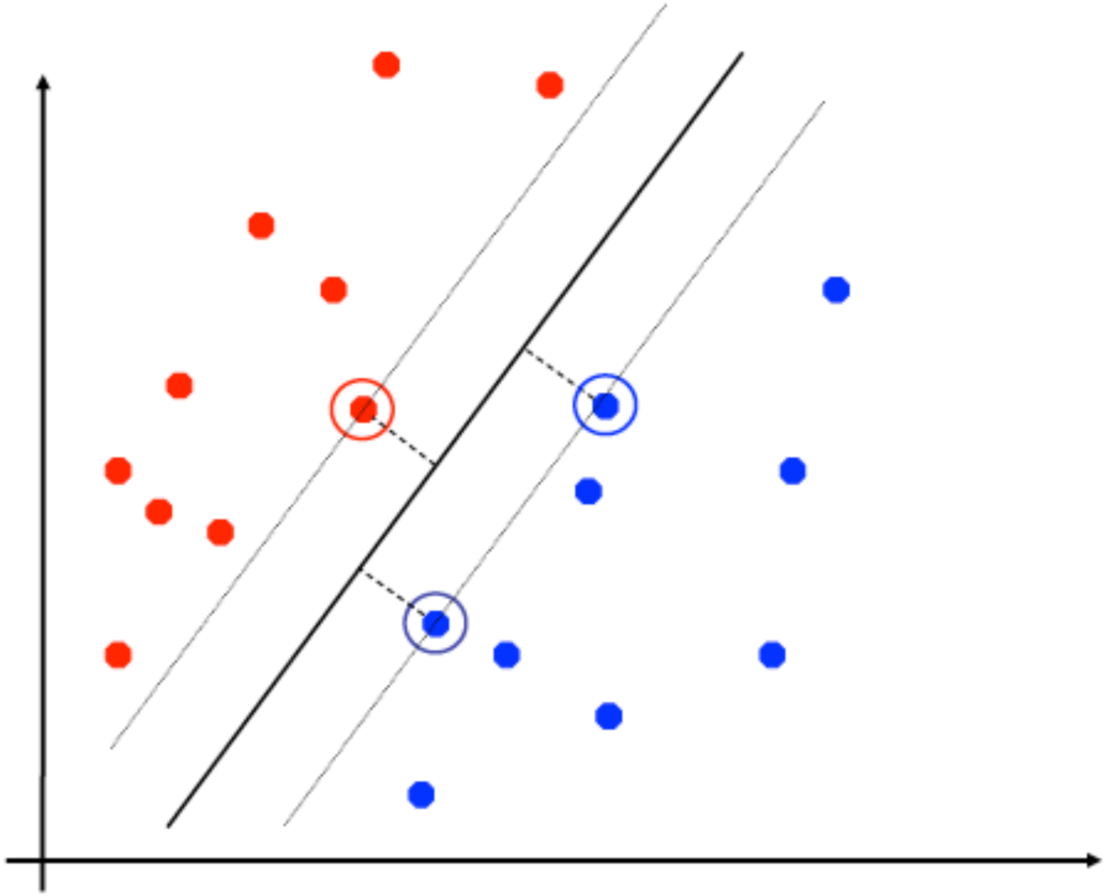
$$\min_{w,b} \sum_j \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$$

The **hinge loss** is defined as $\ell_{\text{hinge}}(y, \hat{y}) = \max\left(0, 1 - \hat{y}y\right)$

$$\min_{\mathbf{w},b} \sum_j \ell_{\text{hinge}}(y_j,\, w \cdot x_j + b)$$

This is empirical risk minimization, using the hinge loss

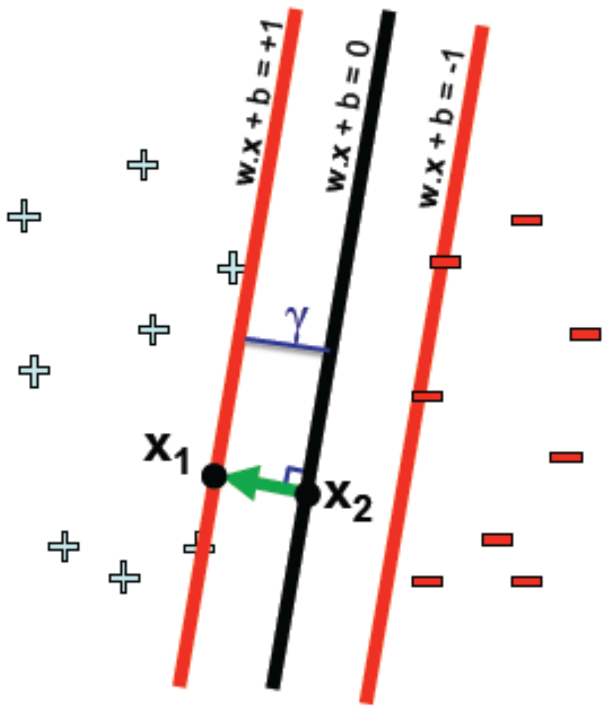
Key idea #2: seek large margin



What is γ (geometric margin) as a function of w?

$\gamma_i = \text{Distance to } i\text{'th data point}$

$\gamma = \min_i \gamma_i$



$$w \cdot x_1 + b = 1$$

-

$$w \cdot x_2 + b = 0$$

$$w \cdot (x_1 - x_2) = 1$$

Plug in

We also know that:

$$x_1 - x_2 = \gamma \frac{w}{||w||}$$

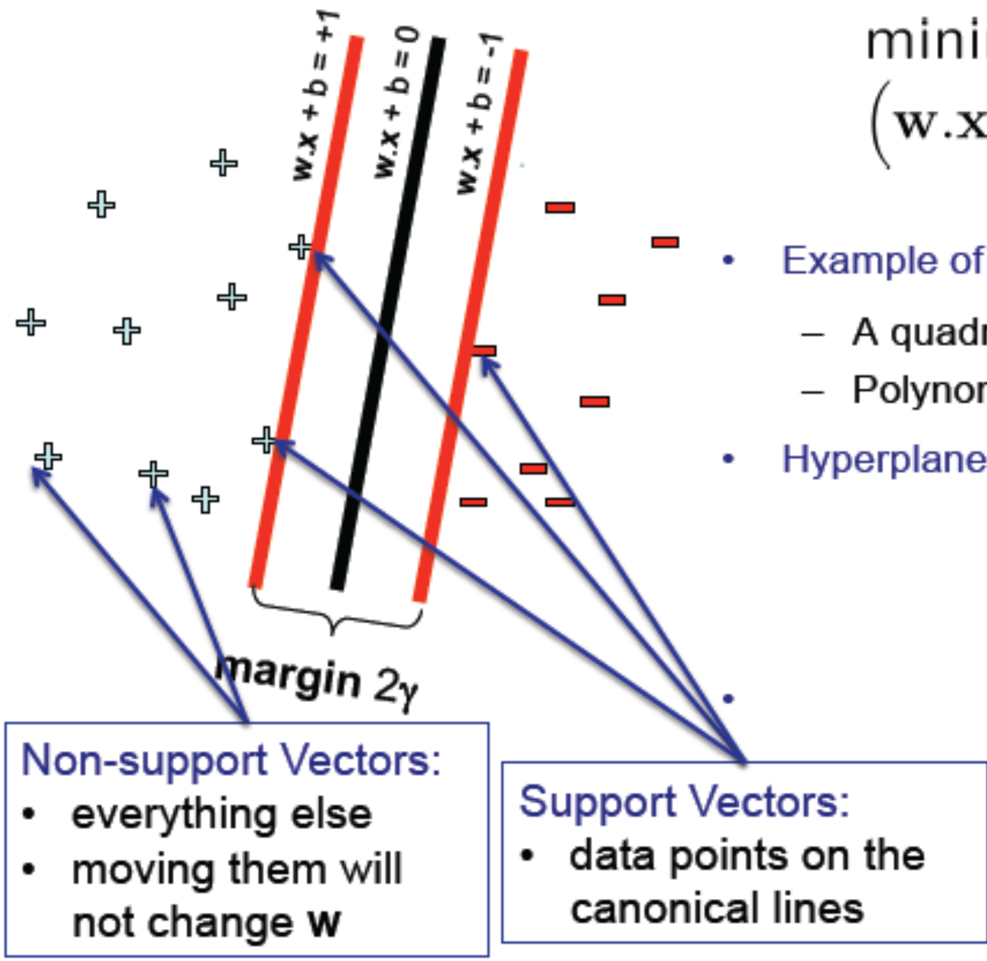
$$1 = w \cdot \left(\gamma \frac{w}{||w||}\right) = \frac{\gamma}{||w||} w \cdot w = \gamma ||w||$$

So, $\gamma = \frac{1}{||w||}$

(assuming there is a data point on the **w.x** + b = +1 or -1 line)

Final result: can maximize γ by minimizing $||w||_2$!!!

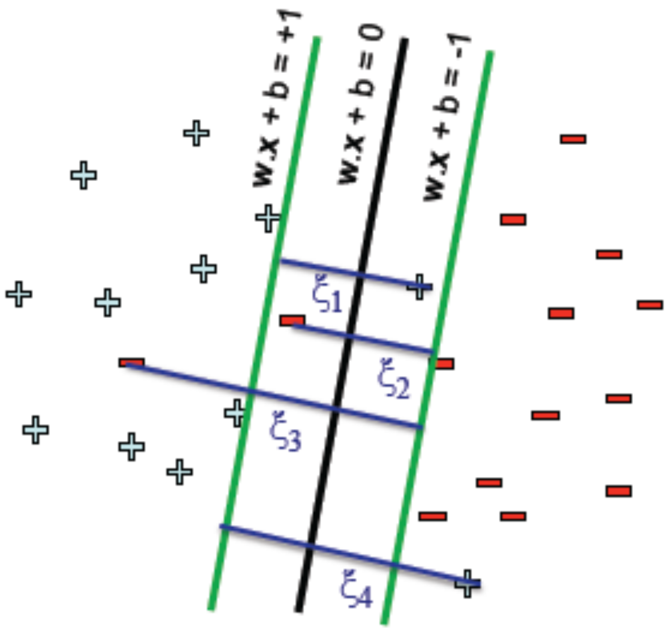
(Hard margin) support vector machines



$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \;\; \forall j$$

- Example of a **convex optimization** problem
  - A quadratic program
  - Polynomial-time algorithms to solve!
- Hyperplane defined by **support vectors**

Allowing for slack: "Soft margin SVM"



$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\, \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

↑ "slack variables"

Slack penalty $C > 0$:

- $C=\infty$ → have to separate the data!
- $C=0$ → ignores the data entirely!

For each data point:

•If margin ≥ 1, don't care

•If margin < 1, pay linear penalty

Equivalent formulation using hinge loss

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\,\xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

Substituting $\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$ into the objective, we get:

$$\min ||w||^2 + C \sum_j \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$$

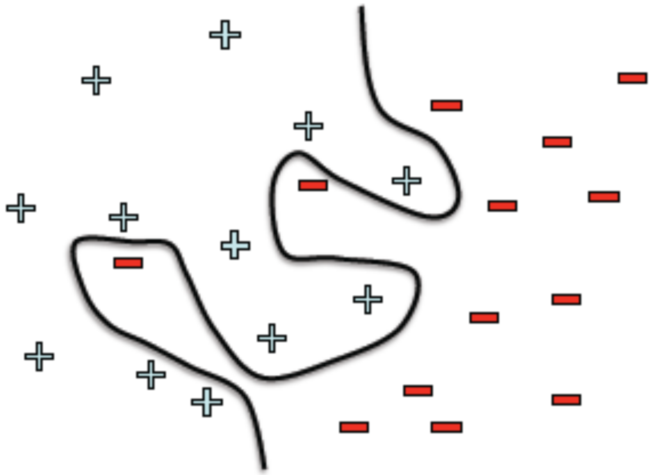
Recall, the **hinge loss** is $\ell_{\text{hinge}}(y, \hat{y}) = \max\Big(0, 1 - \hat{y}y\Big)$

$$\min_{\mathbf{w},b} ||w||_2^2 + C \sum_j \ell_{\text{hinge}}(y_j,\, w \cdot x_j + b)$$

This is called **regularization**; used to prevent overfitting!

This part is empirical risk minimization, using the hinge loss

What if the data is not linearly separable?

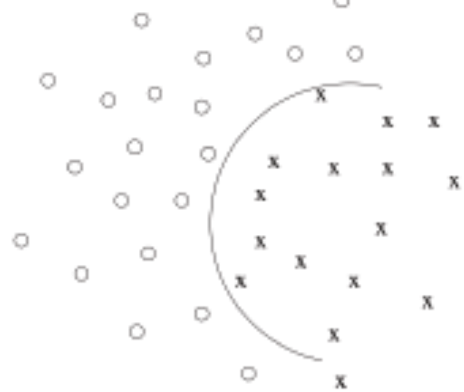


Use features of features of features of features….

$$\phi(x) = \begin{pmatrix} x^{(1)} \\ \ldots \\ x^{(n)} \\ x^{(1)}x^{(2)} \\ x^{(1)}x^{(3)} \\ \ldots \\ e^{x^{(1)}} \\ \ldots \end{pmatrix}$$

Feature space can get really large really quickly!

$$\varphi(x_1,x_2) \rightarrow (x_1^{\,2}, x^1x^2, x^2x^1, x_2^{\,2})$$



Non-linear separator in the original $\mathbf{x}$-space

Linear separator in the feature ϕ-space

[Tommi Jaakkola]

Key idea #3: the kernel trick

- High dimensional feature spaces at no extra cost!

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- As a result, prediction can be performed with:

$$\begin{aligned}\hat{y} &\leftarrow \mathrm{sign}(\mathbf{w} \cdot \phi(\mathbf{x})) \\ &= \mathrm{sign}\Big((\sum_i \alpha_i y_i \phi(\mathbf{x}_i)) \cdot \phi(\mathbf{x}) \Big) \\ &= \mathrm{sign}\Big(\sum_i \alpha_i y_i (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})) \Big) \\ &= \mathrm{sign}\Big(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \Big) \quad \text{where } K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}').\end{aligned}$$

Key idea #3: the kernel trick

- High dimensional feature spaces at no extra cost!

Kernel method enables one to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space but rather by simply computing the inner products between the images of all pairs of data in the feature space.

Often computationally cheaper than the explicit computation of the coordinates.

Polynomial kernel

d=1

$$\phi(u).\phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} . \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = u.v$$

d=2

$$\phi(u).\phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} . \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix} = u_1^2 v_1^2 + 2 u_1 v_1 u_2 v_2 + u_2^2 v_2^2$$
$$= (u_1 v_1 + u_2 v_2)^2$$
$$= (u.v)^2$$

For any *d (we will skip proof):*

$$\phi(u).\phi(v) = (u.v)^d$$

Polynomials of degree **exactly** *d*

Common kernels

- Polynomials of degree exactly *d*

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to *d*

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Gaussian kernels

$$K(\vec{u},\vec{v}) = \exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$$

- And many others: very active area of research!