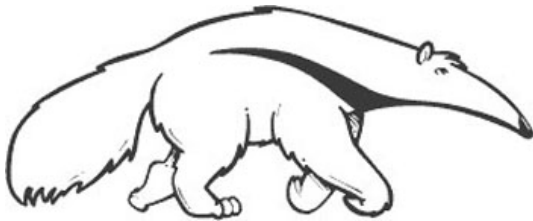


CS178: Machine Learning and Data Mining

Support Vector Machines

Prof. Erik Sudderth



Some materials courtesy Alex Ihler & Sameer Singh

CS178 Zoom Lectures

CS178 [zoom](#) lectures are recorded by the instructor (the recording feature is disabled for students). Recordings are posted to [YuJa](#), and only available to CS178 students and staff. To ask questions during lecture, you may:

- Use the **Raise Hand** feature. Prof. Sudderth will then call on you by name, unmute your microphone, and let you ask a question. *Your question will be recorded. Please be respectful of your instructor and classmates.*
- Use the **Q&A Window** to type a question. Prof. Sudderth will read your question to the class before answering it, but *will not personally identify you.*

Machine Learning

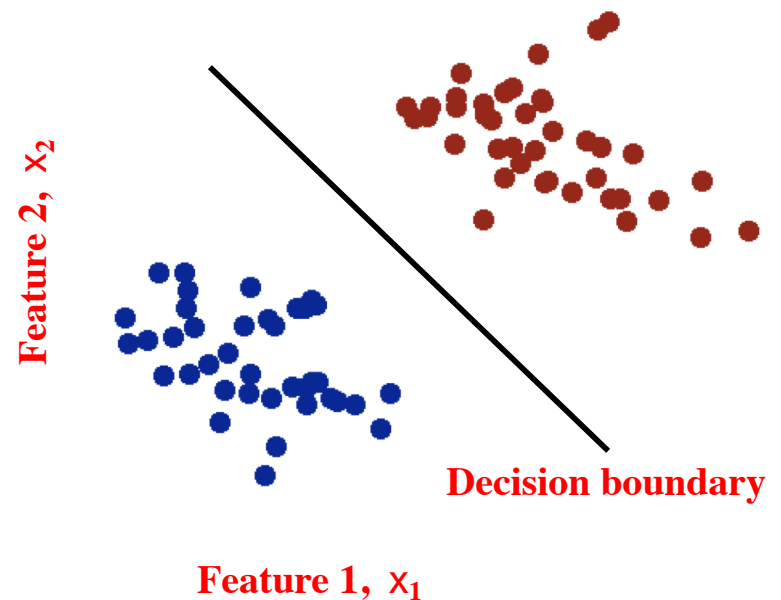
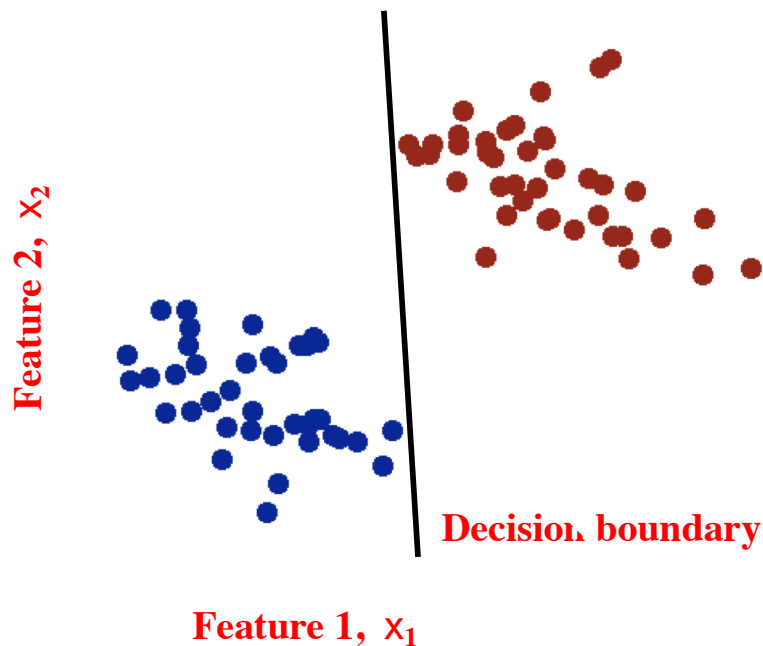
Support Vector Machines

Lagrangian and Dual

The Kernel Trick

Linear classifiers

- Which decision boundary is “better”?
 - Both have zero training error (perfect training accuracy)
 - But, one of them seems intuitively better...
- How can we quantify “better”,
and learn the “best” parameter settings?



One possible answer...

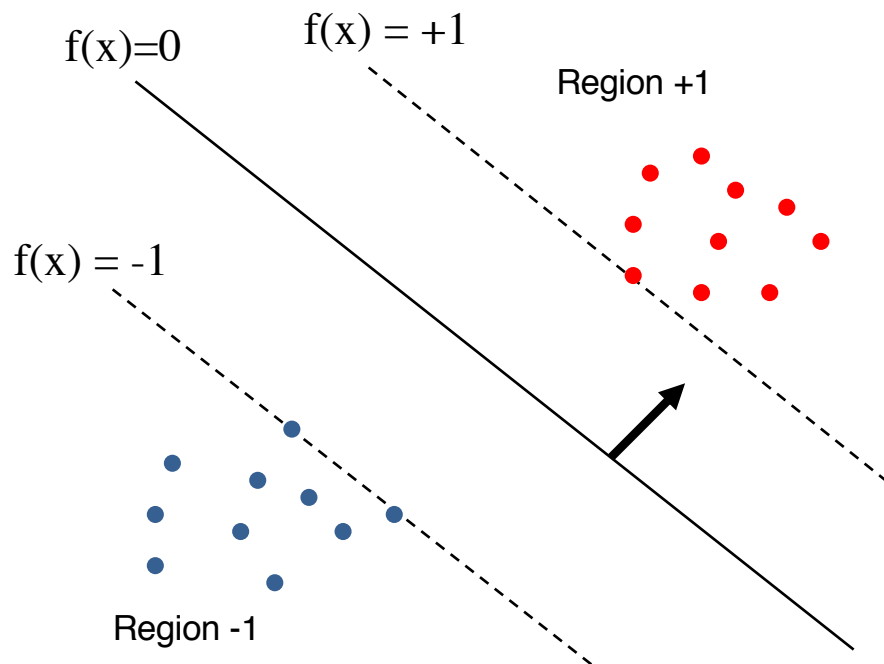
- Maybe we want to maximize our “margin”
- To optimize, relate to model parameters
- Remove “scale invariance”
 - Define class +1 in some region, class –1 in another
 - Make those regions as far apart as possible

Notation change!

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$$



$$b + w_1 x_1 + w_2 x_2 + \dots$$



We could define such a function:

$$f(x) = w * x' + b$$

$$f(x) > +1 \text{ in region +1}$$

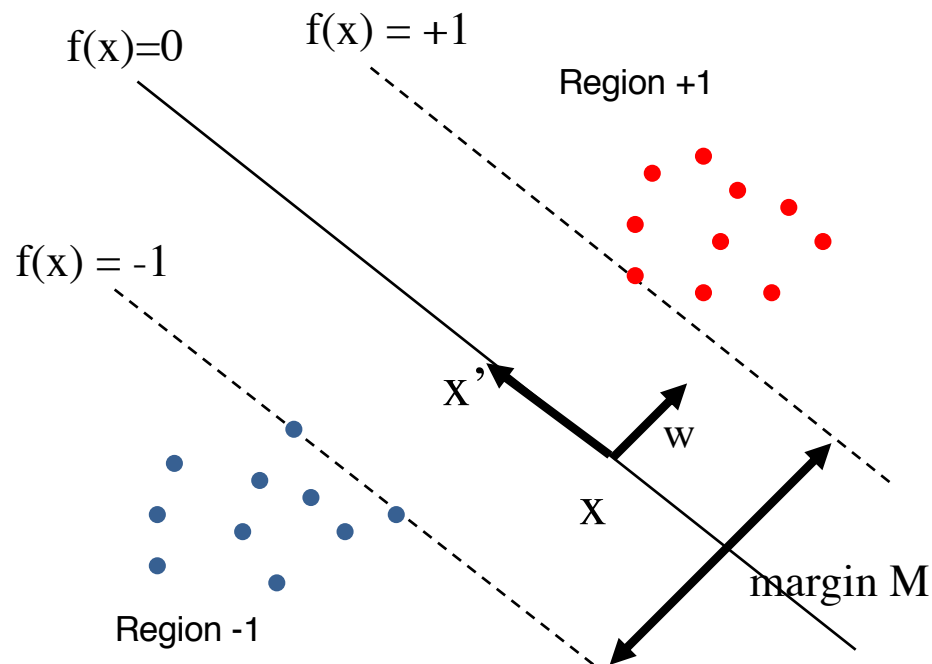
$$f(x) < -1 \text{ in region -1}$$

Passes through zero in center...

“Support vectors” – data points on margin

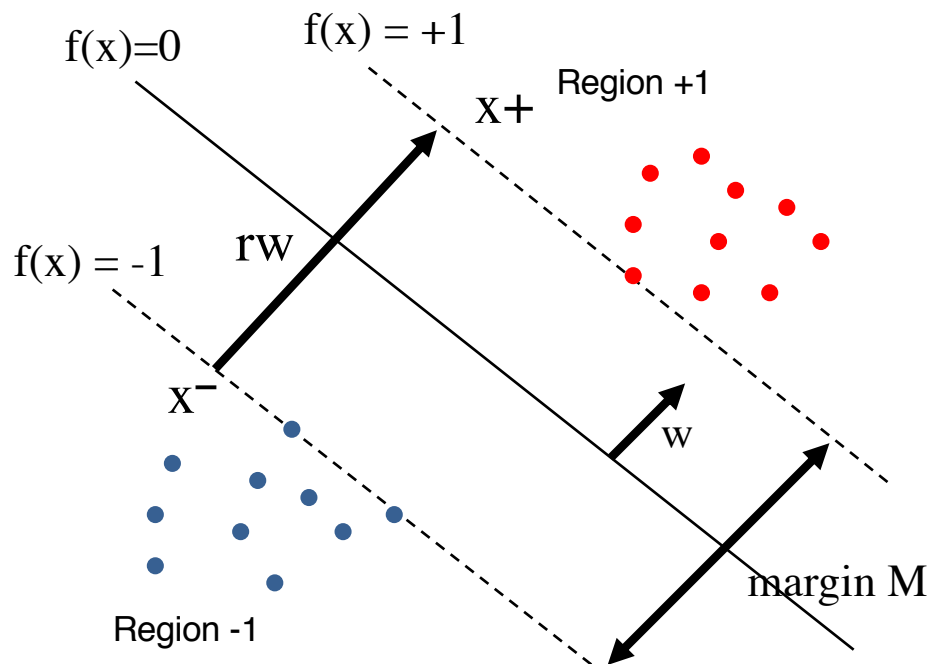
Computing the margin width

- Vector $\underline{w}=[w_1 \ w_2 \ \dots]$ is perpendicular to the boundaries (why?)
- $w x + b = 0$ & $w x' + b = 0 \Rightarrow w (x' - x) = 0$: orthogonal



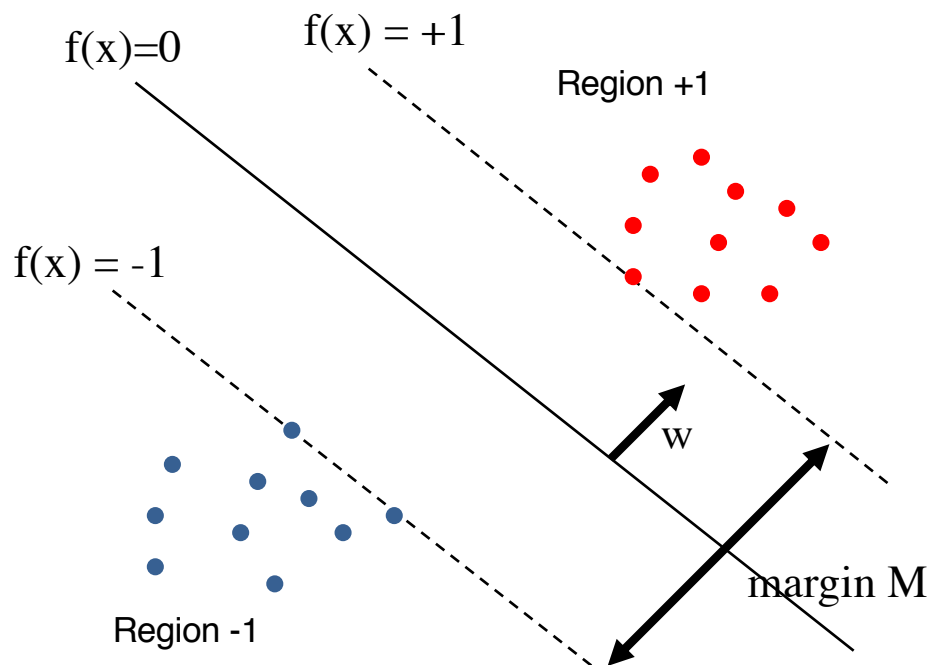
Computing the margin width

- Vector $\underline{w}=[w_1 \ w_2 \ \dots]$ is perpendicular to the boundaries
- Choose \underline{x}^- st $f(\underline{x}^-) = -1$; let \underline{x}^+ be the closest point with $f(\underline{x}^+) = +1$
 - $\underline{x}^+ = \underline{x}^- + r * \underline{w}$ (why?)
- Closest two points on the margin also satisfy
$$w \cdot x^- + b = -1 \qquad w \cdot x^+ + b = +1$$



Computing the margin width

- Vector $\underline{w}=[w_1 \ w_2 \ \dots]$ is perpendicular to the boundaries
- Choose \underline{x}^- st $f(\underline{x}^-) = -1$; let \underline{x}^+ be the closest point with $f(\underline{x}^+) = +1$
 - $\underline{x}^+ = \underline{x}^- + r * \underline{w}$
- Closest two points on the margin also satisfy
$$w \cdot x^- + b = -1 \qquad w \cdot x^+ + b = +1$$



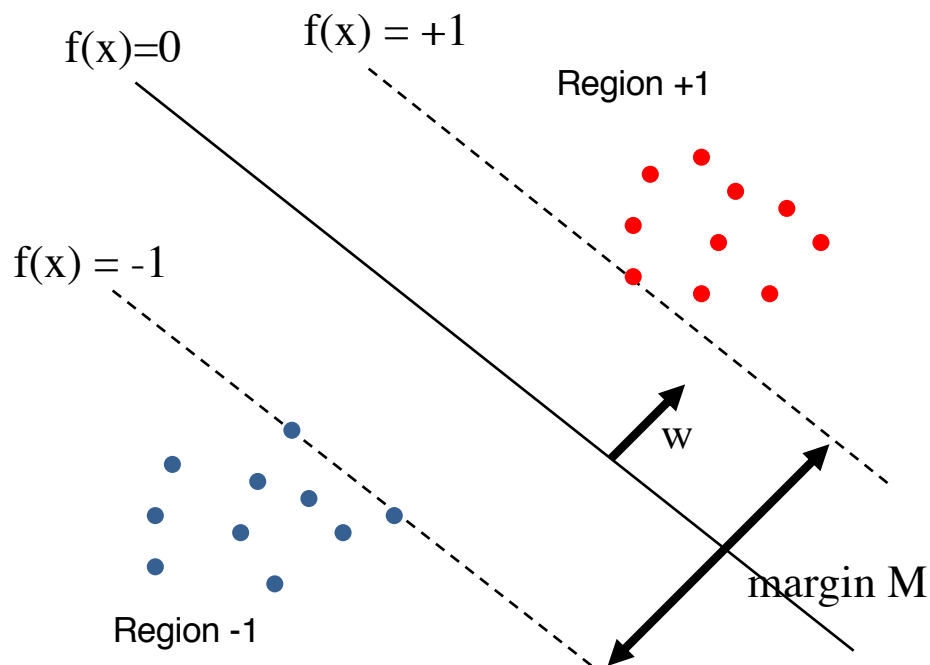
$$\begin{aligned} w \cdot (x^- + rw) + b &= +1 \\ \Rightarrow r\|w\|^2 + w \cdot x^- + b &= +1 \\ \Rightarrow r\|w\|^2 - 1 &= +1 \\ \Rightarrow r &= \frac{2}{\|w\|^2} \end{aligned}$$

$$\begin{aligned} M &= \|x^+ - x^-\| = \|rw\| \\ &= \frac{2}{\|w\|^2} \|w\| = \frac{2}{\sqrt{w^T w}} \end{aligned}$$

Maximum margin classifier

- Constrained optimization
 - Get all data points correct
 - Maximize the margin

This is an example of a quadratic program:
quadratic cost function, linear constraints



$$w^* = \arg \max_w \frac{2}{\sqrt{w^T w}}$$

such that “all data on the correct side of the margin”

Primal problem:

$$w^* = \arg \min_w \sum_j w_j^2$$

s.t.

$$y^{(i)} = +1 \Rightarrow w \cdot x^{(i)} + b \geq +1$$

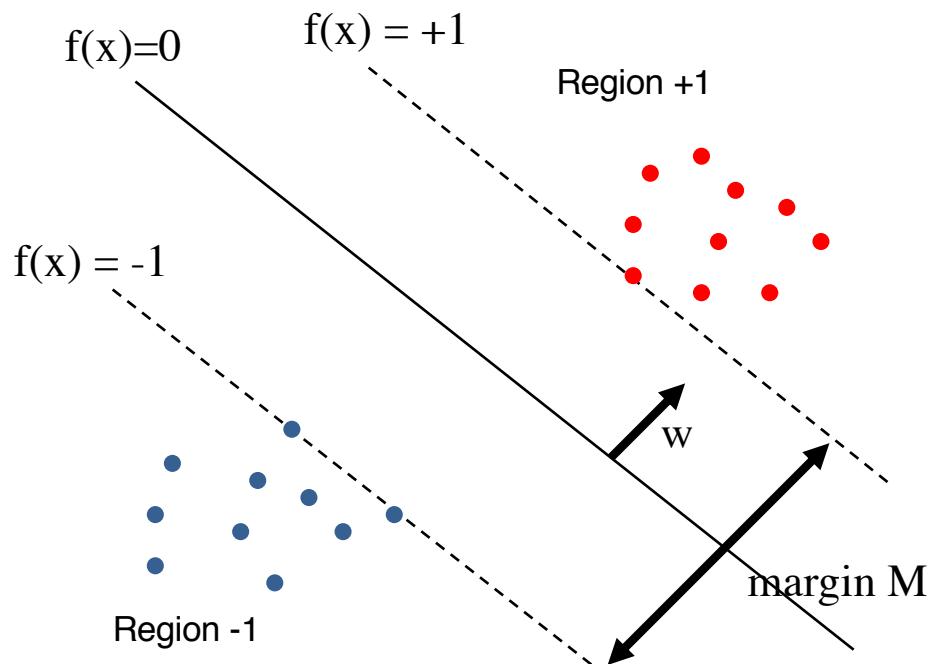
$$y^{(i)} = -1 \Rightarrow w \cdot x^{(i)} + b \leq -1$$

(m constraints)

Maximum margin classifier

- Constrained optimization
 - Get all data points correct
 - Maximize the margin

This is an example of a quadratic program:
quadratic cost function, linear constraints



$$w^* = \arg \max_w \frac{2}{\sqrt{w^T w}}$$

such that “all data on the correct side of the margin”

Primal problem:

$$w^* = \arg \min_w \sum_j w_j^2$$

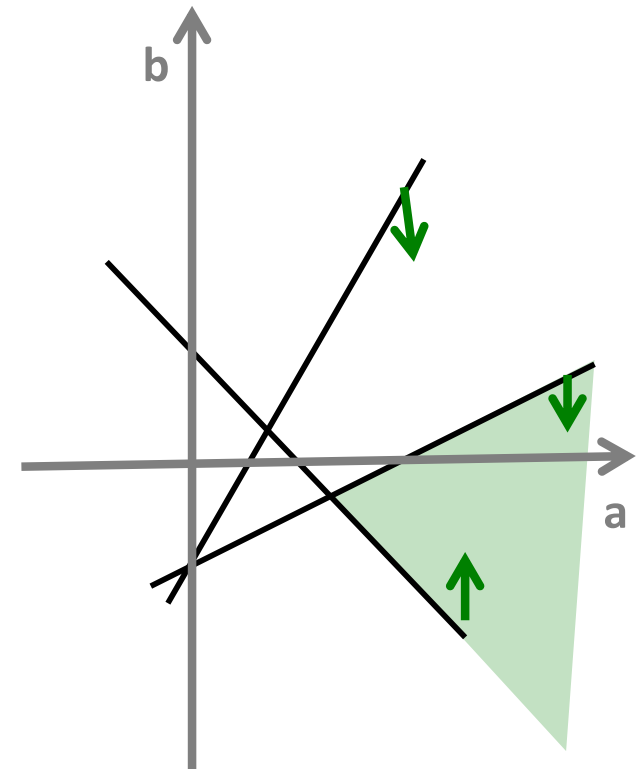
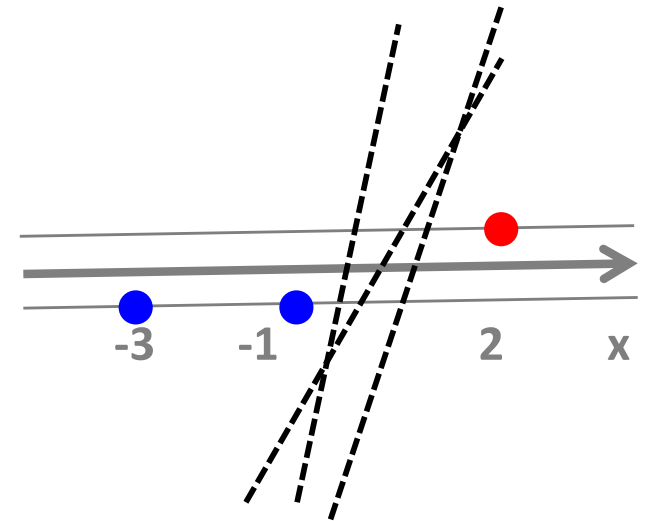
s.t.

$$y^{(i)} (w \cdot x^{(i)} + b) \geq +1$$

(m constraints)

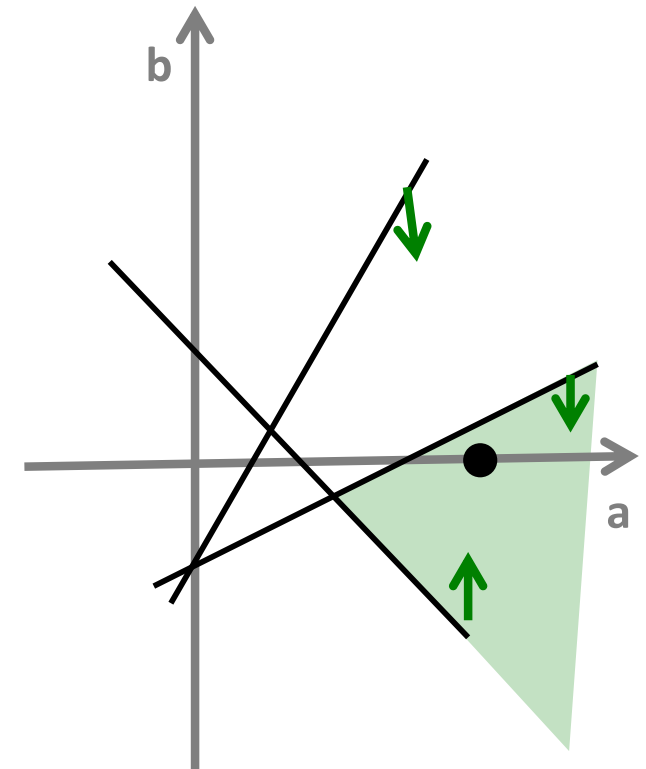
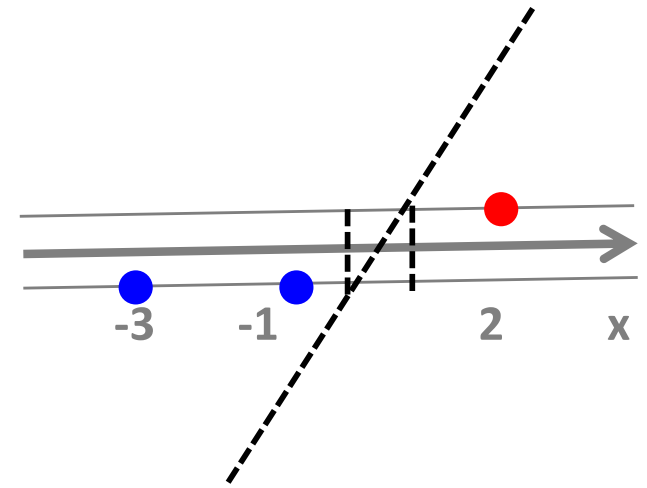
A 1D Example

- Suppose we have three data points
 $x = -3, y = -1$
 $x = -1, y = -1$
 $x = 2, y = 1$
- Many separating perceptrons, $T[ax+b]$
 - Anything with $ax+b = 0$ between -1 and 2
- We can write the margin constraints
 $a(-3) + b < -1 \quad \Rightarrow b < 3a - 1$
 $a(-1) + b < -1 \quad \Rightarrow b < a - 1$
 $a(2) + b > +1 \quad \Rightarrow b > -2a + 1$



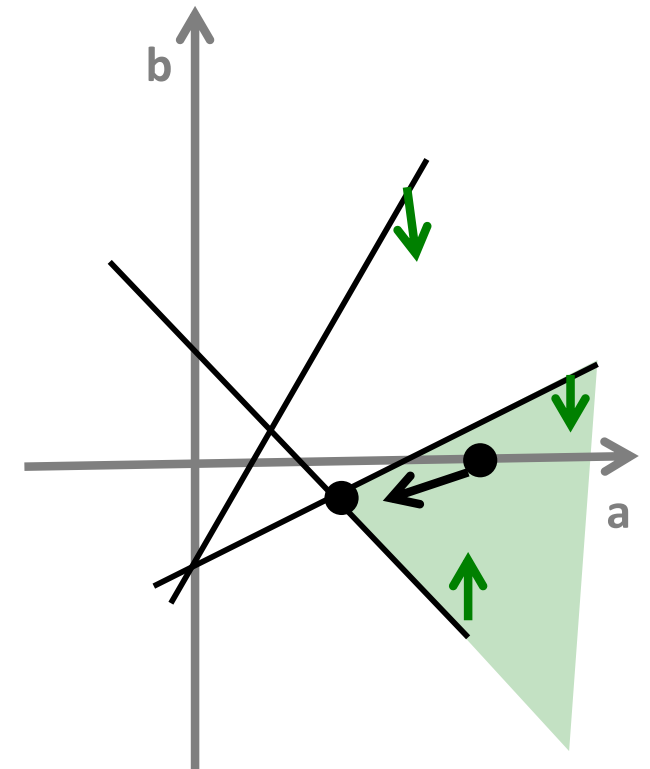
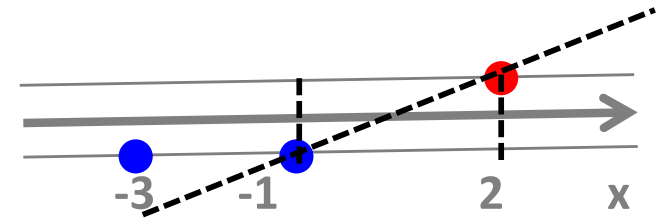
A 1D Example

- Suppose we have three data points
 $x = -3, y = -1$
 $x = -1, y = -1$
 $x = 2, y = 1$
- Many separating perceptrons, $T[ax+b]$
 - Anything with $ax+b = 0$ between -1 and 2
- We can write the margin constraints
 $a(-3) + b < -1 \quad \Rightarrow b < 3a - 1$
 $a(-1) + b < -1 \quad \Rightarrow b < a - 1$
 $a(2) + b > +1 \quad \Rightarrow b > -2a + 1$
- Ex: $a = 1, b = 0$



A 1D Example

- Suppose we have three data points
 - $x = -3, y = -1$
 - $x = -1, y = -1$
 - $x = 2, y = 1$
- Many separating perceptrons, $T[ax+b]$
 - Anything with $ax+b = 0$ between -1 and 2
- We can write the margin constraints
 - $a(-3) + b < -1 \Rightarrow b < 3a - 1$
 - $a(-1) + b < -1 \Rightarrow b < a - 1$
 - $a(2) + b > +1 \Rightarrow b > -2a + 1$
- Ex: $a = 1, b = 0$
- Minimize $||a|| \Rightarrow a = .66, b = -.33$
 - Two data on the margin; constraints “tight”



Machine Learning

Support Vector Machines

Lagrangian and Dual

The Kernel Trick

Lagrangian optimization

- Want to optimize constrained system:

$\theta = (w, b)$

$$w^* = \arg \min_{w, b} \underbrace{\sum_j w_j^2}_{f(\theta)} \quad s.t. \quad \underbrace{1 - y^{(i)}(w \cdot x^{(i)} + b)}_{g_i(\theta) \leq 0} \leq 0$$

- Introduce Lagrange multipliers α (one per constraint)

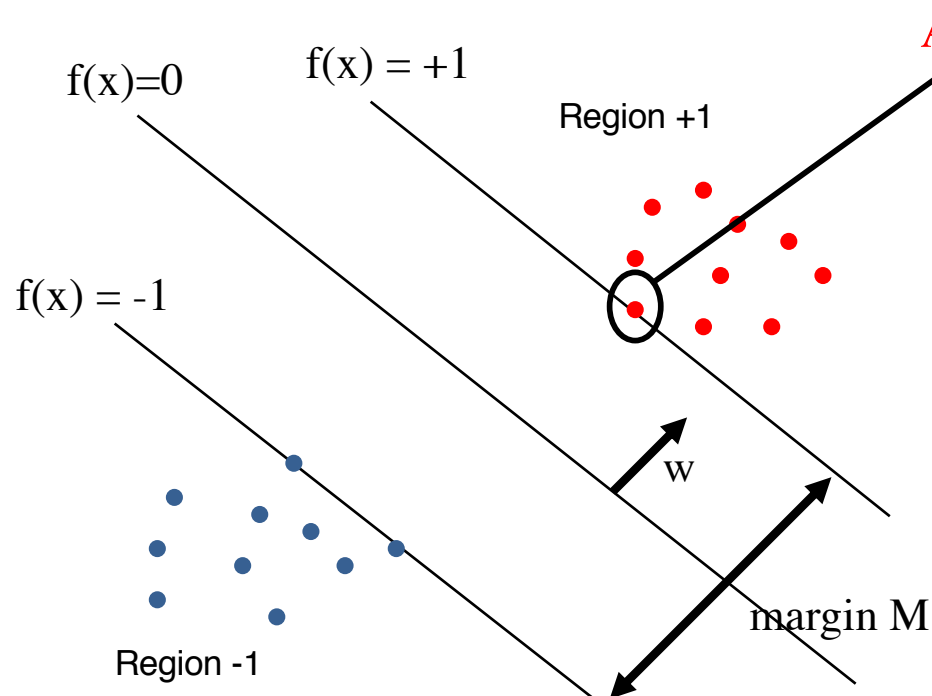
$$\theta^* = \arg \min_{\theta} \max_{\alpha \geq 0} \underbrace{f(\theta)}_{\text{red}} + \sum_i \alpha_i \underbrace{g_i(\theta)}_{\text{blue}}$$

- Can optimize θ, α jointly over a simpler constraint set (initialization easy)
- For inner max:
 - $g_i(\theta) < 0 : \alpha_i = 0$
 - $g_i(\theta) > 0 : \alpha_i \rightarrow +\infty$
- Any optimum of the original problem is a saddle point of the new
- KKT complementary slackness: $\alpha_i > 0 \Rightarrow g_i(\theta) = 0$

Optimization

- Use Lagrange multipliers
 - Enforce inequality constraints

$$w^* = \arg \min_w \max_{\alpha \geq 0} \frac{1}{2} \sum_j w_j^2 + \sum_i \alpha_i (1 - y^{(i)} (w \cdot x^{(i)} + b))$$



Alphas > 0 only on the margin:
“support vectors”

Stationary conditions wrt w :

$$w^* = \sum_i \alpha_i y^{(i)} x^{(i)}$$

and since any support vector has $y = wx + b$,

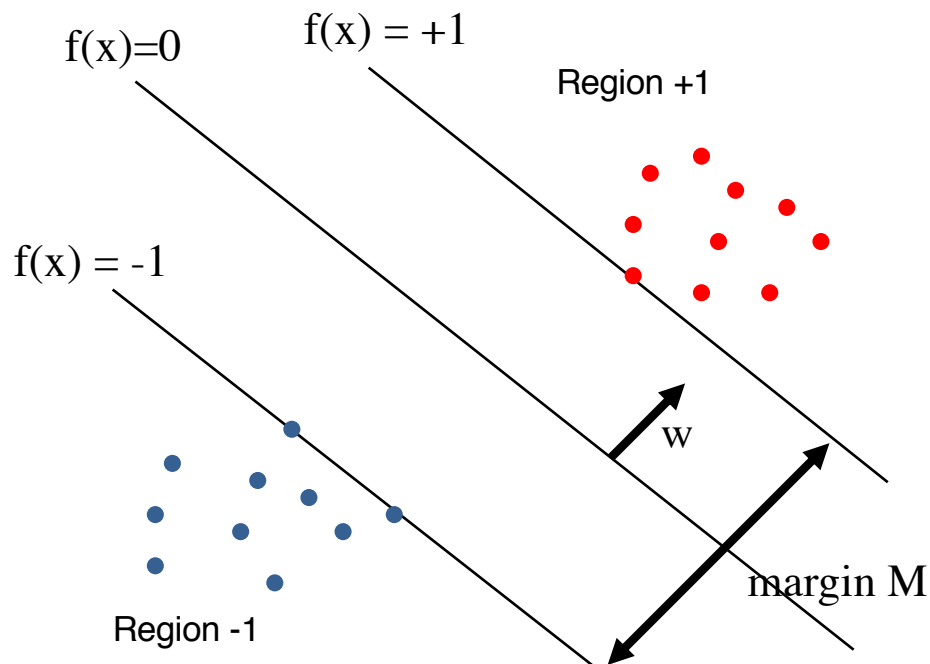
$$b = \frac{1}{N_{sv}} \sum_{i \in SV} (y^{(i)} - w \cdot x^{(i)})$$

Dual form

- Use Lagrange multipliers
 - Enforce inequality constraints
 - Use solution w^* to write solely in terms of alphas:

$$\max_{\alpha \geq 0} \sum_i \left[\alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)}) \right]$$

$$\text{s.t. } \sum_i \alpha_i y^{(i)} = 0 \quad (\text{since derivative wrt } b = 0)$$



Another quadratic program:

optimize m vars with $1+m$ (simple) constraints
cost function has m^2 dot products

$$w^* = \sum_i \alpha_i y^{(i)} x^{(i)}$$

$$b = \frac{1}{N_{sv}} \sum_{i \in SV} (y^{(i)} - w \cdot x^{(i)})$$

Maximum margin classifier

- What if the data are not linearly separable?

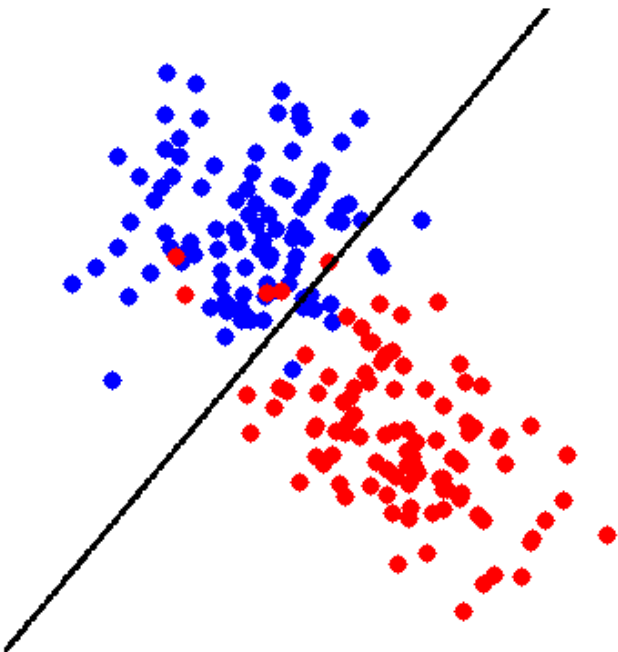
- Want a large “margin”:

$$\min_w \sum_j w_j^2$$

- Want low error:

$$\min_w \sum_i J(y^{(i)}, w \cdot x^{(i)} + b)$$

- “Soft margin” : introduce slack variables for violated constraints



$$w^* = \arg \min_{w, \epsilon} \sum_j w_j^2 + R \sum_i \epsilon^{(i)}$$

s.t.

$$y^{(i)} (w^T x^{(i)} + b) \geq +1 - \epsilon^{(i)} \quad (\text{violate margin by } \epsilon)$$
$$\epsilon^{(i)} \geq 0$$

Assigns “cost” R proportional to distance from margin
Another quadratic program!

Soft margin SVM

- Large margin vs. Slack variables
- R large = hard margin
- R smaller
 - A few wrong predictions; boundary farther from rest

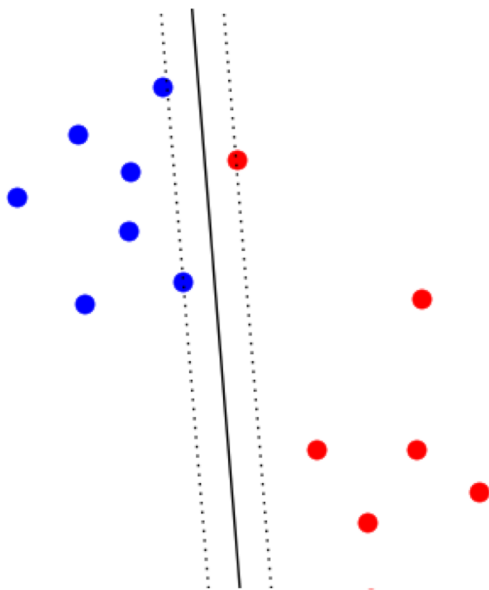
$$w^* = \arg \min_{w, \epsilon} \sum_j w_j^2 + R \sum_i \epsilon^{(i)}$$

s.t.

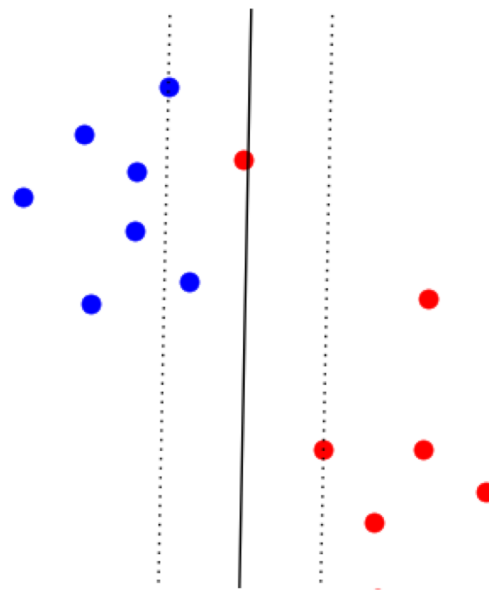
$$y^{(i)}(w^T x^{(i)} + b) \geq +1 - \epsilon^{(i)}$$

$$\epsilon^{(i)} \geq 0$$

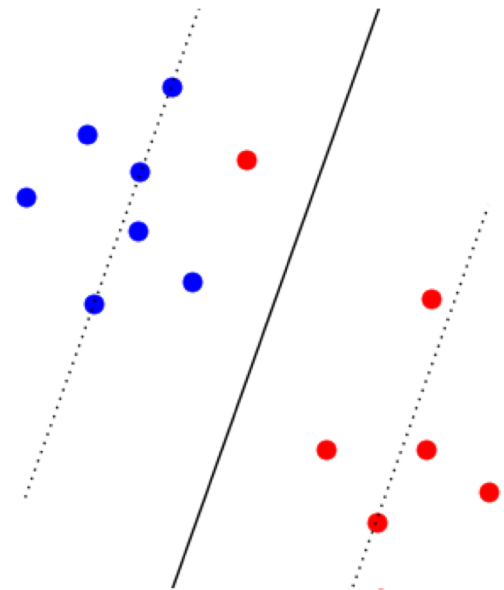
$R = R_0$



$R = 10^{-2} R_0$



$R = 10^{-4} R_0$



Maximum margin classifier

- Soft margin optimization:
 - For *any* weights w , we can choose ϵ to satisfy constraints

$$w^* = \arg \min_{w, \epsilon} \sum_j w_j^2 + R \sum_i \epsilon^{(i)}$$

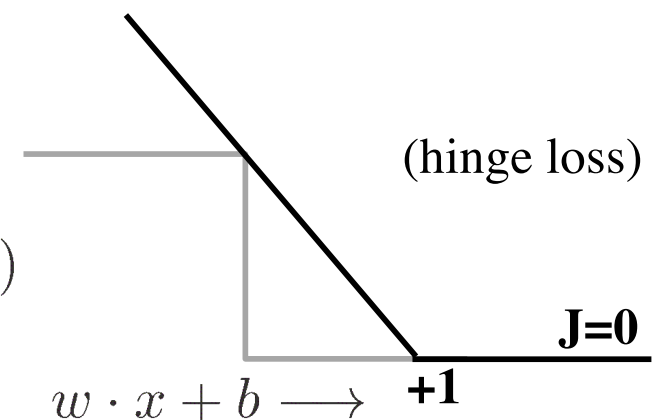
$$y^{(i)} (w^T x^{(i)} + b) \geq +1 - \epsilon^{(i)}$$

- Write ϵ^* as a function of w (call this J) and optimize directly
- J = distance from the “correct” place

$$J_i = \max[0, 1 - y^{(i)} (w \cdot x^{(i)} + b)]$$

$$w^* = \arg \min_w \frac{1}{R} \sum_j w_j^2 + \sum_i J_i(y^{(i)}, w \cdot x^{(i)} + b)$$

(L2 regularization on the weights)



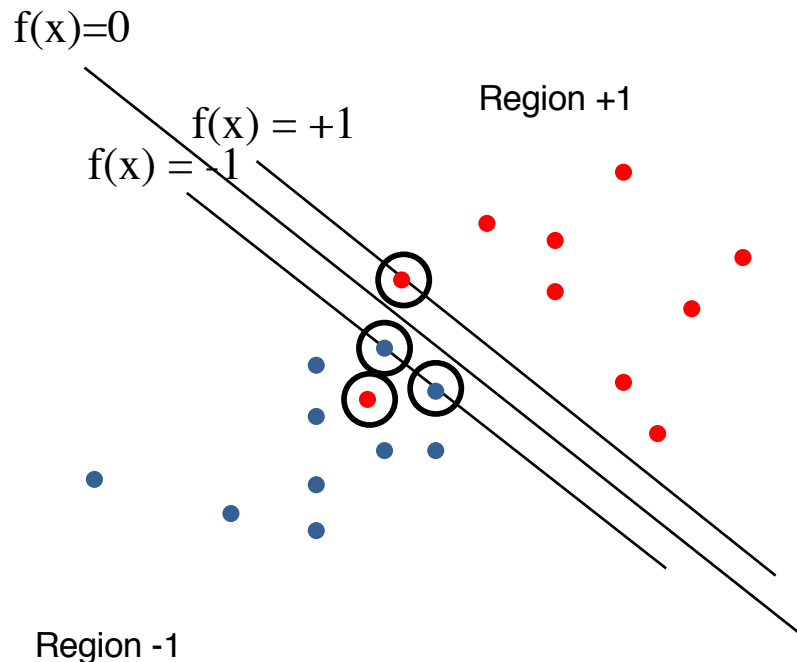
Dual form

- Soft margin dual:

$$\max_{\underline{0 \leq \alpha \leq R}} \sum_i \alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \underbrace{x^{(i)} \cdot x^{(j)}}_{K_{ij}}$$

K_{ij} measures “similarity” of x_i and x_j (their dot product)

$$\text{s.t. } \sum_i \alpha_i y^{(i)} = 0$$



Support vectors now data on or past margin...

Prediction:

$$\hat{y} = w^* \cdot x + b = \sum_i \alpha_i y^{(i)} \underbrace{x^{(i)} \cdot x}_{K_{ij}} + b$$

$$w^* = \sum_i \alpha_i y^{(i)} x^{(i)}$$

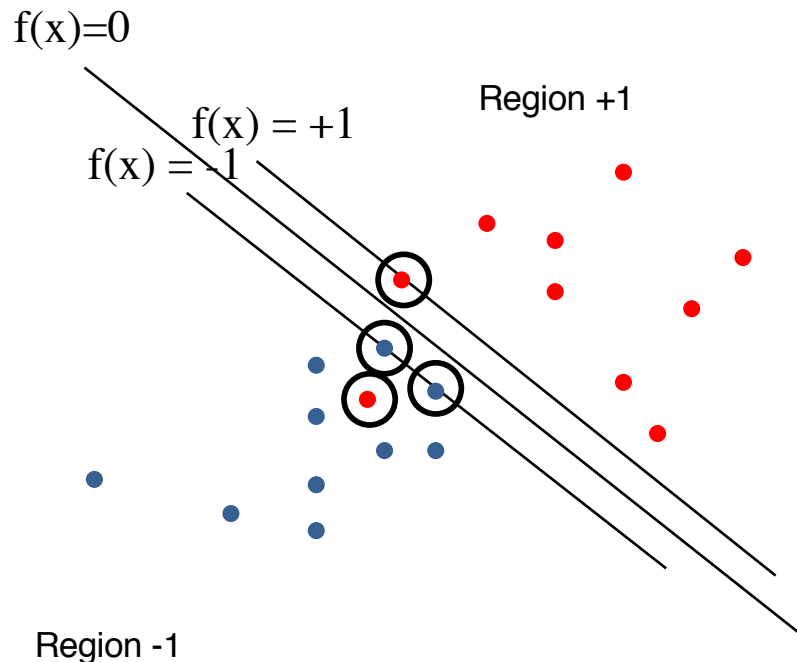
$b = \dots$ More complicated; can solve e.g. using any $\alpha \in (0, R)$

Support Vectors

The *support vectors* are data points i with non-zero weight α_i :

- Points with minimum margin (on optimized boundary)
- Points which violate margin constraint, but are still correctly classified
- Points which are misclassified

For all other training data, features have *no impact* on learned weight vector



Support vectors now data on or past margin...

Prediction:

$$\hat{y} = w^* \cdot x + b = \sum_i \alpha_i y^{(i)} x^{(i)} \cdot x + b$$

$$w^* = \sum_i \alpha_i y^{(i)} x^{(i)}$$

$b = \dots$ More complicated; can solve
e.g. using any $\alpha \in (0, R)$

Machine Learning

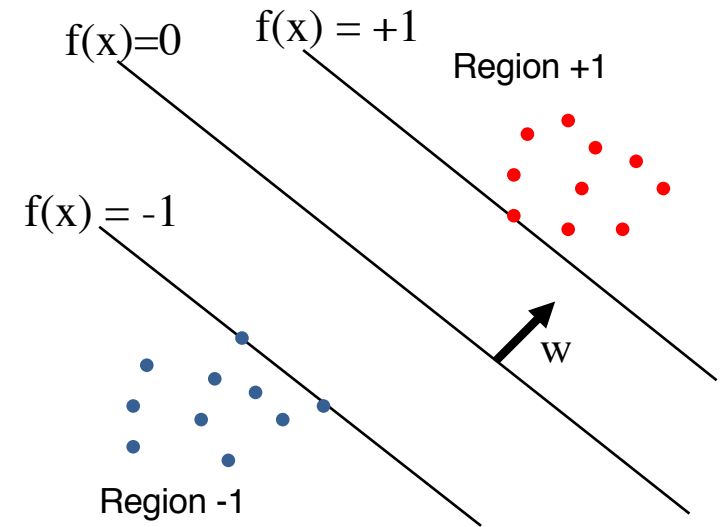
Support Vector Machines

Lagrangian and Dual

The Kernel Trick

Linear SVMs

- So far, looked at linear SVMs:
 - Expressible as linear weights “w”
 - Linear decision boundary



- Dual optimization for a linear SVM:

$$\max_{0 \leq \alpha \leq R} \sum_i \alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)}) \quad \text{s.t.} \quad \sum_i \alpha_i y^{(i)} = 0$$

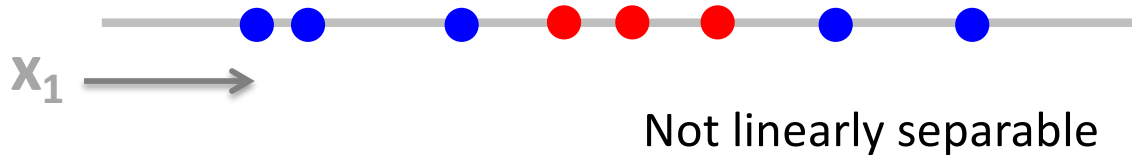
- Depend on pairwise dot products:

- K_{ij} measures “similarity”, e.g., 0 if orthogonal $K_{ij} = x^{(i)} \cdot x^{(j)}$

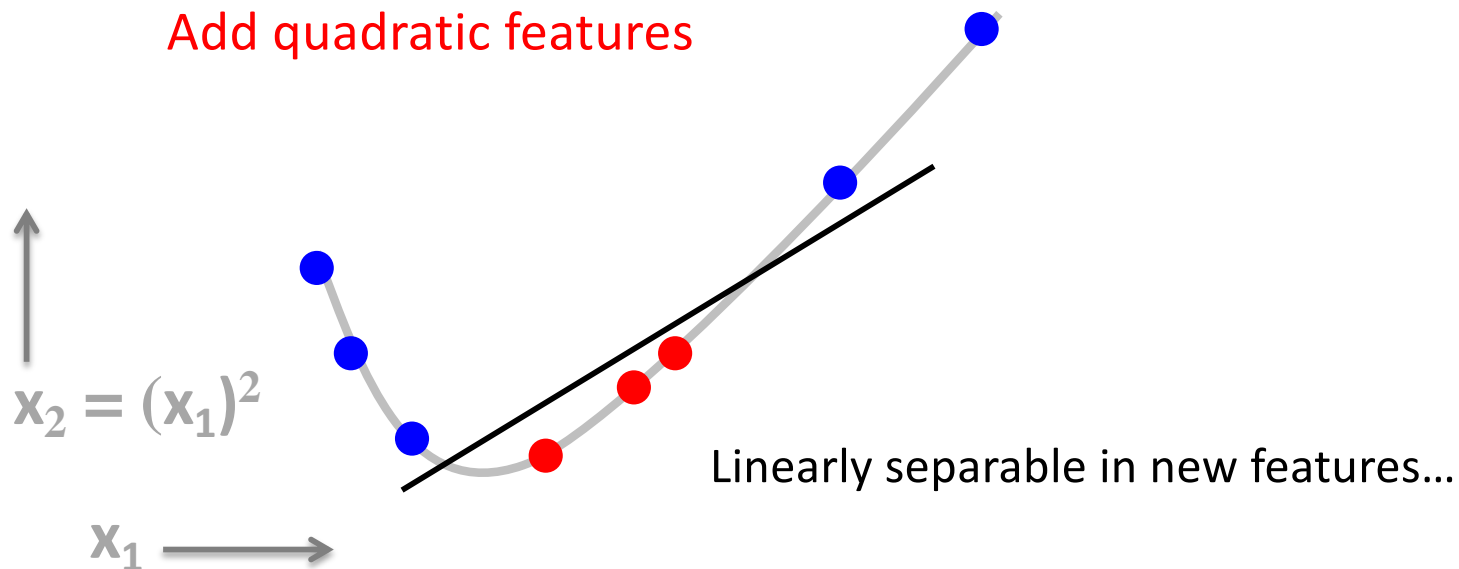
Adding features

- Linear classifier can't learn some functions

1D example:



Add quadratic features



Adding features

- Recall: feature function $\Phi(x)$
 - Predict using some transformation of original features

$$\hat{y}(x) = \text{sign}[w \cdot \Phi(x) + b]$$

- Dual form of SVM optimization is:

$$\max_{0 \leq \alpha \leq R} \sum_i \alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)}) \Phi(x^{(j)})^T \quad \text{s.t.} \quad \sum_i \alpha_i y^{(i)} = 0$$

- For example, quadratic (polynomial) features:

$$\Phi(x) = (1 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad \cdots \quad x_1^2 \quad x_2^2 \quad \cdots \quad \sqrt{2}x_1x_2 \quad \sqrt{2}x_1x_3 \quad \cdots)$$

- Ignore root-2 scaling for now...
- Expands “x” to length $O(n^2)$

Implicit features

- Need $\Phi(x^{(i)})\Phi(x^{(j)})^T$

$$\Phi(x) = (1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ \cdots \ x_1^2 \ x_2^2 \ \cdots \ \sqrt{2}x_1x_2 \ \sqrt{2}x_1x_3 \ \cdots)$$

$$\Phi(a) = (1 \ \sqrt{2}a_1 \ \sqrt{2}a_2 \ \cdots \ a_1^2 \ a_2^2 \ \cdots \ \sqrt{2}a_1a_2 \ \sqrt{2}a_1a_3 \ \cdots)$$

$$\Phi(b) = (1 \ \sqrt{2}b_1 \ \sqrt{2}b_2 \ \cdots \ b_1^2 \ b_2^2 \ \cdots \ \sqrt{2}b_1b_2 \ \sqrt{2}b_1b_3 \ \cdots)$$

$$\Phi(a)^T \Phi(b) = 1 + \sum_j 2a_j b_j + \sum_j a_j^2 b_j^2 + \sum_j \sum_{k>j} 2a_j a_k b_j b_k + \dots$$

$$= (1 + \sum_j a_j b_j)^2$$

$$= K(a, b)$$

Can evaluate dot product in
only $O(n)$ computations!

Mercer Kernels

- If $K(x, x')$ satisfies Mercer's condition:

$$\int_a \int_b K(a, b) g(a) g(b) da db \geq 0$$

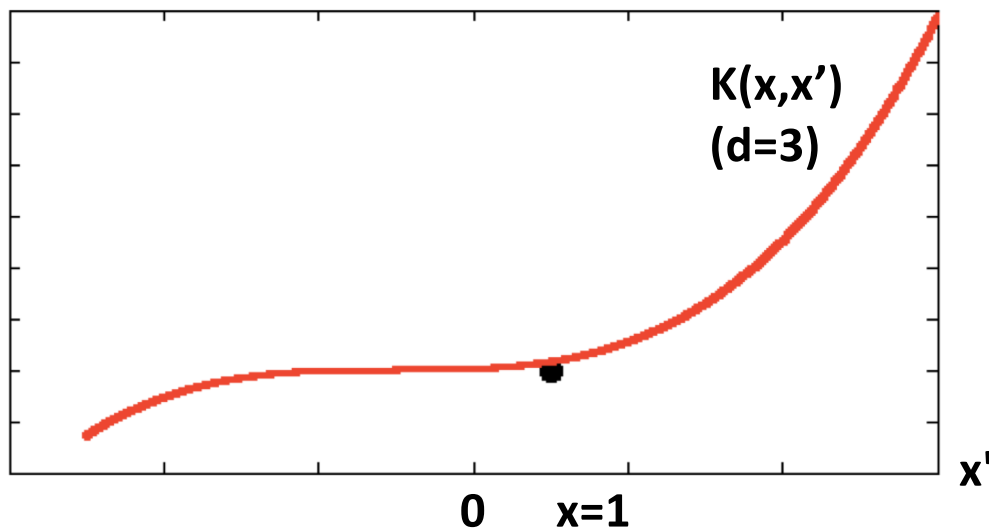
For all datasets X :

$$g^T \cdot K \cdot g \geq 0$$

- Then, $K(a, b) = \Phi(a) \cdot \Phi(b)$ for some $\Phi(x)$
- Notably, Φ may be hard to calculate
 - May even be infinite dimensional!
 - Only matters that $K(x, x')$ is easy to compute:
 - Computation always stays $O(m^2)$

Common kernel functions

- Some commonly used kernel functions & their shape:
- Polynomial $K(a, b) = (1 + \sum_j a_j b_j)^d$



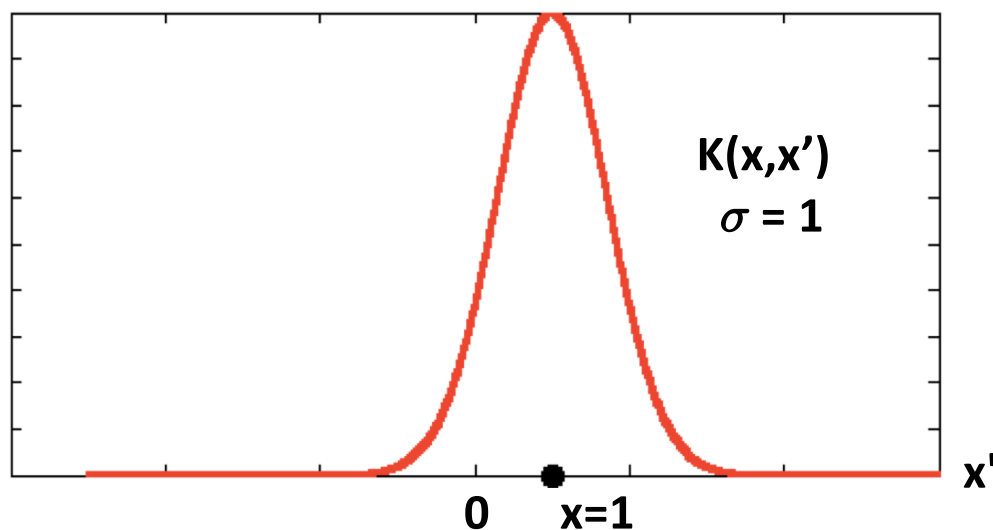
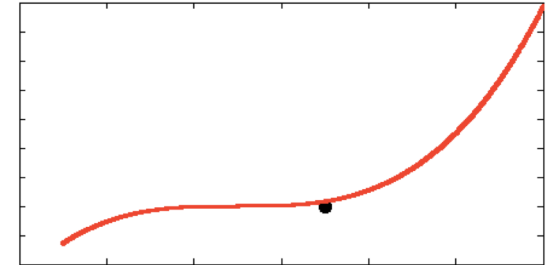
Common kernel functions

- Some commonly used kernel functions & their shape:

- Polynomial $K(a, b) = (1 + \sum_j a_j b_j)^d$

- Radial Basis Functions

$$K(a, b) = \exp(-(a - b)^2 / 2\sigma^2)$$



Common kernel functions

- Some commonly used kernel functions & their shape:

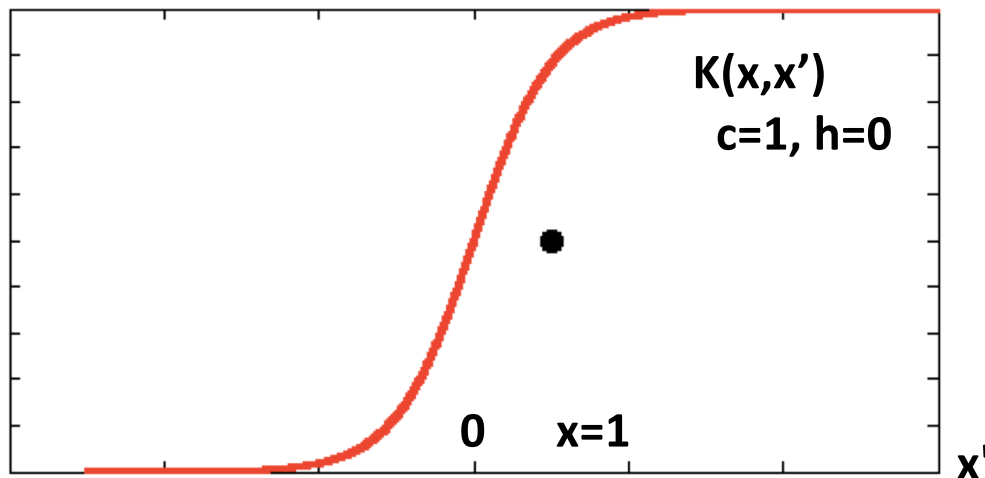
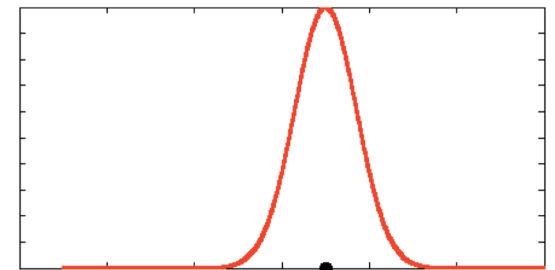
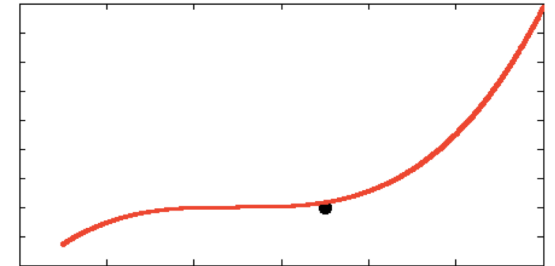
- Polynomial $K(a, b) = (1 + \sum_j a_j b_j)^d$

- Radial Basis Functions

$$K(a, b) = \exp(-(a - b)^2 / 2\sigma^2)$$

- Saturating, sigmoid-like:

$$K(a, b) = \tanh(ca^T b + h)$$



Common kernel functions

- Some commonly used kernel functions & their shape:

- Polynomial $K(a, b) = (1 + \sum_j a_j b_j)^d$

- Radial Basis Functions

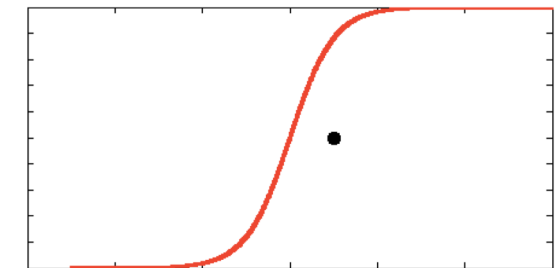
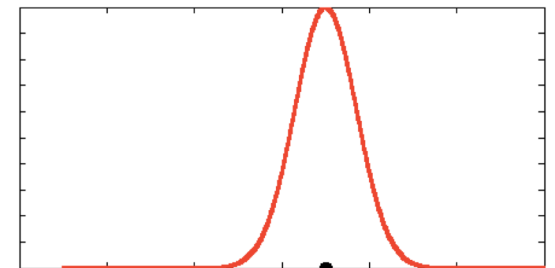
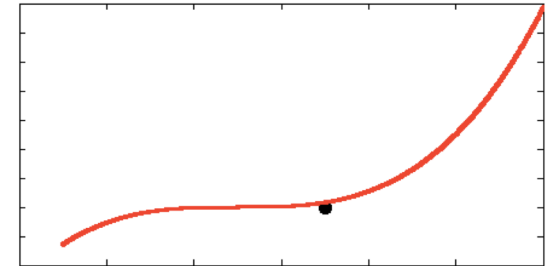
$$K(a, b) = \exp(-(a - b)^2 / 2\sigma^2)$$

- Saturating, sigmoid-like:

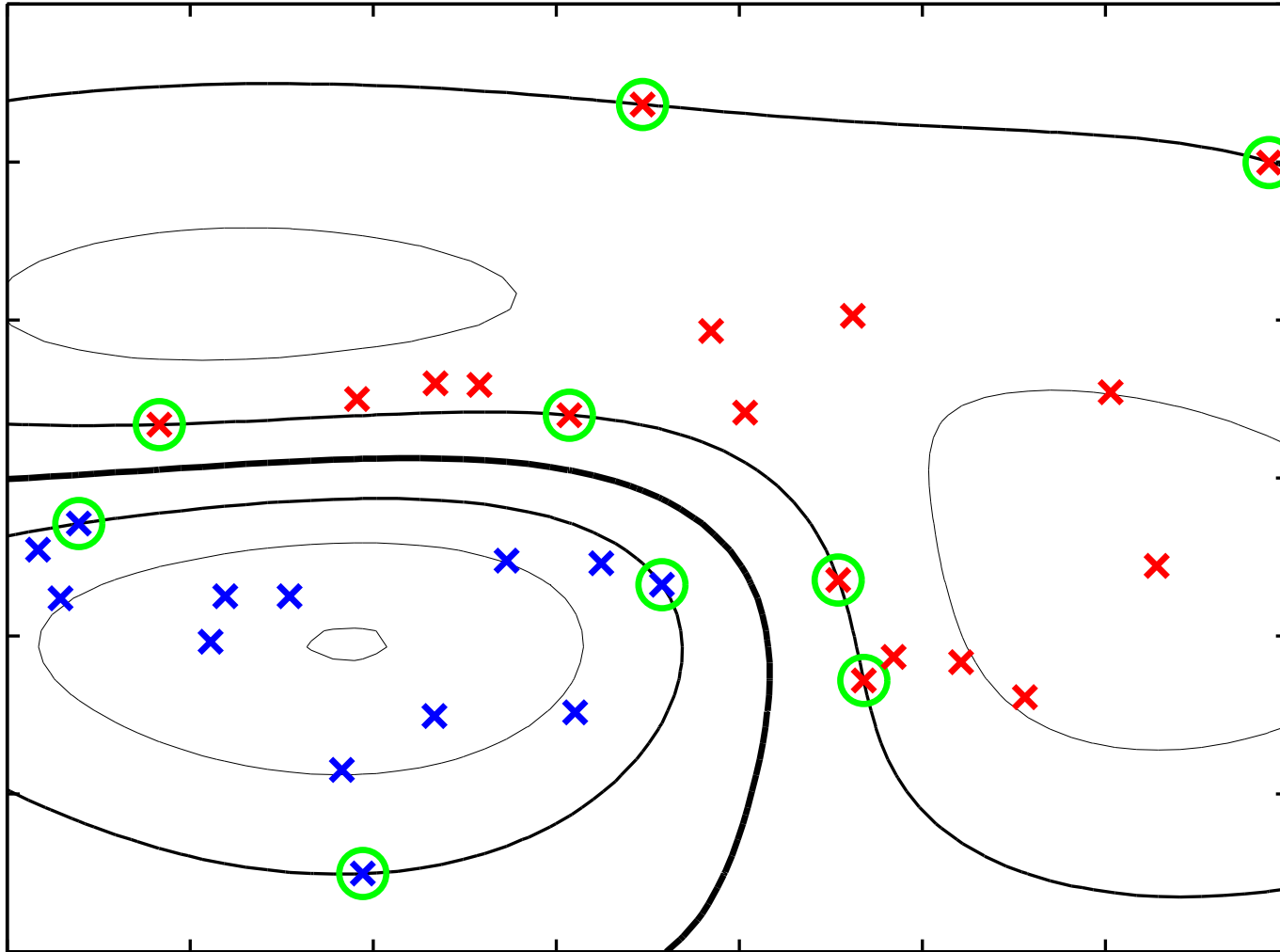
$$K(a, b) = \tanh(ca^T b + h)$$

- Many for special data types:
 - String similarity for text, genetics

- In practice, may not even be Mercer kernels...

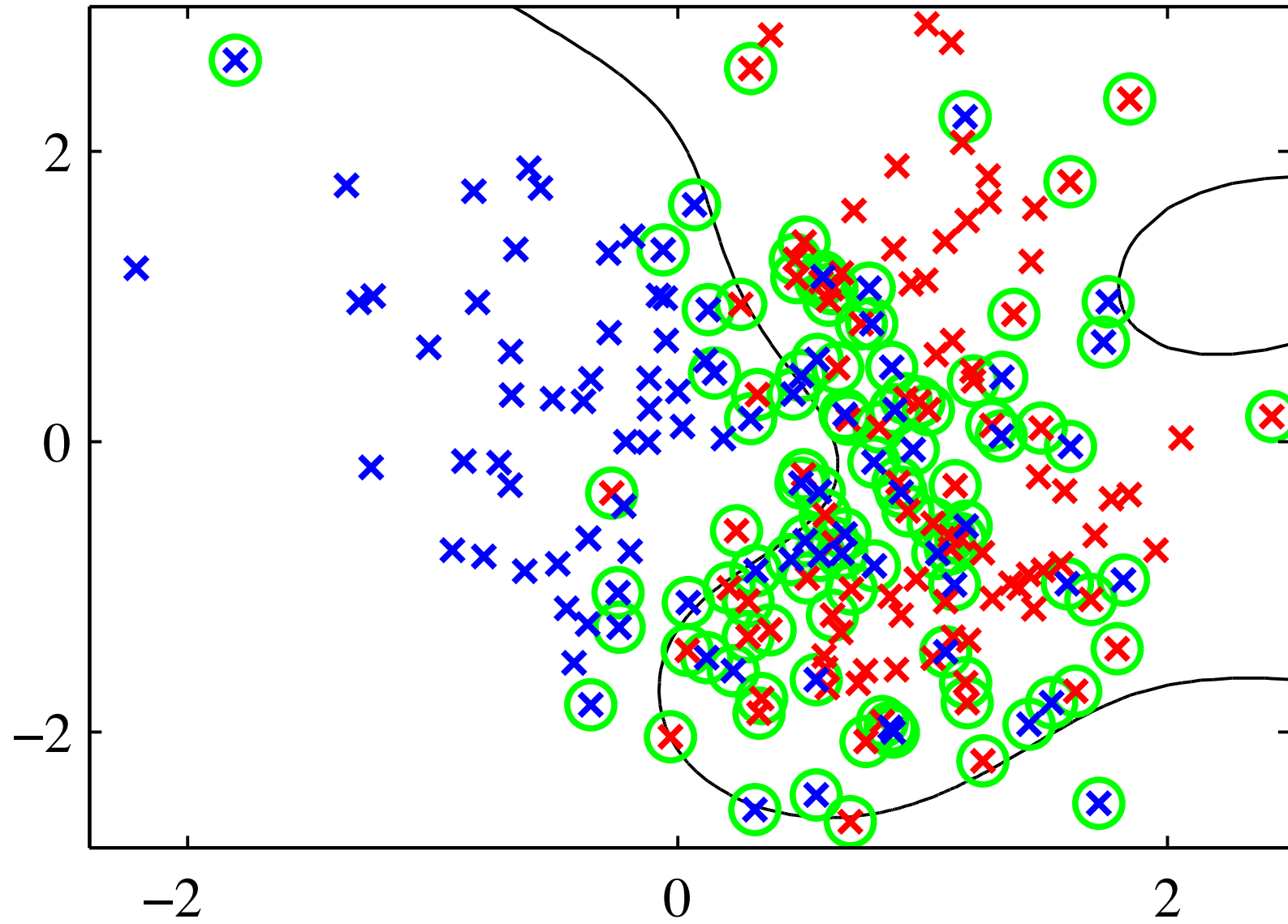


Support Vectors for Kernel SVMs



Support vectors (green) for data separable by radial basis function kernels, and non-linear margin boundaries

How Many Support Vectors?



*Only need to evaluate kernel at support vectors, not all training data.
But there may still be a lot of support vectors.*

Kernel SVMs

- Linear SVMs
 - Can represent classifier using $(w,b) = n+1$ parameters
 - Or, represent using support vectors, $x^{(i)}$
- Kernelized?
 - $K(x,x')$ may correspond to high (infinite?) dimensional $\Phi(x)$
 - Typically more efficient to remember the SVs
 - “Instance based” – save data, rather than parameters
- Contrast:
 - Linear SVM: identify *features* with linear relationship to target
 - Kernel SVM: identify *similarity measure* between data

(Sometimes one may be easier; sometimes the other!)

Summary

- Support vector machines
- “Large margin” for separable data
 - Primal QP: maximize margin subject to linear constraints
 - Lagrangian optimization simplifies constraints
 - Dual QP: m variables; involves m^2 dot product
- “Soft margin” for non-separable data
 - Primal form: regularized hinge loss
 - Dual form: m -dimensional QP
- Kernels
 - Dual form involves only pairwise similarity
 - Mercer kernels: dot products in implicit high-dimensional space