The Kernel Trick

Adopted from slides by Alexander Ihler

Primal and dual problem

Example

Primal

$$
\begin{array}{llr}
Max\ 40x_1 + 30x_2 & & (profits) \\
s.t. \quad x_1 + \ \ x_2 \le 120 & & (land) \\
\quad\quad 4x_1 + 2x_2 \le 320 & & (labor) \\
\quad\quad x_1, \quad x_2 \ge 0 & &
\end{array}
$$

Dual

$$
\begin{array}{llr}
\quad\quad (land) \quad (labor) & & \\
Min\ 120y_1 + 320y_2 & & \\
s.t. \quad y_1 + \ \ 4y_2 \ge 40 & & (x_1) \\
\quad\quad y_1 + \ \ 2y_2 \ge 30 & & (x_2) \\
\quad\quad y_1, \quad y_2 \ge 0 & &
\end{array}
$$

SVM with hard constraints

- Primal form

$$w^* = \arg\min_{w,b} \sum_j w_j^2$$

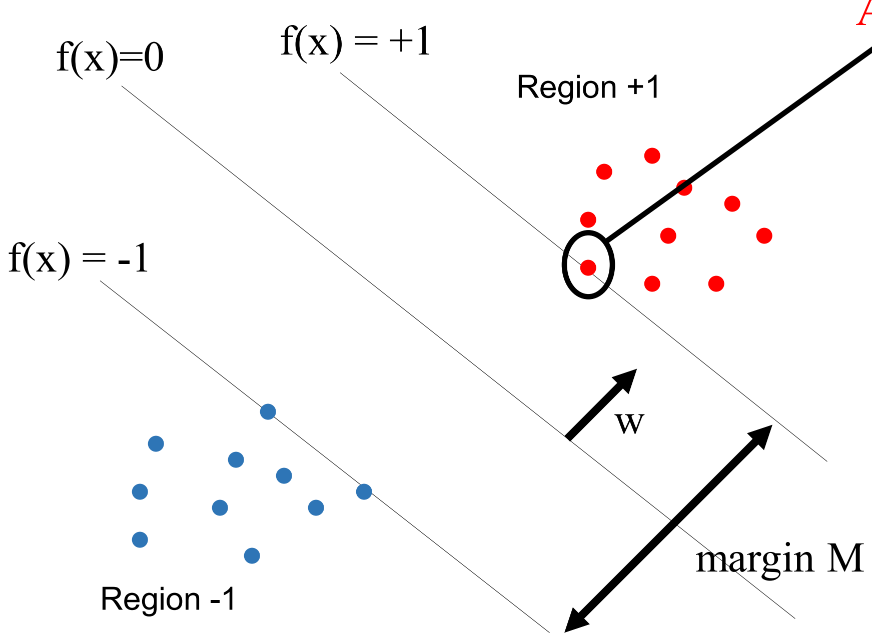
$$s.t. \qquad y^{(i)}(\, w \cdot x^{(i)} + b\,) \geq +1$$

- Dual form

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

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

Support Vectors



Alphas > 0 only on the margin:
"support vectors"

Stationary conditions wrt w:

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

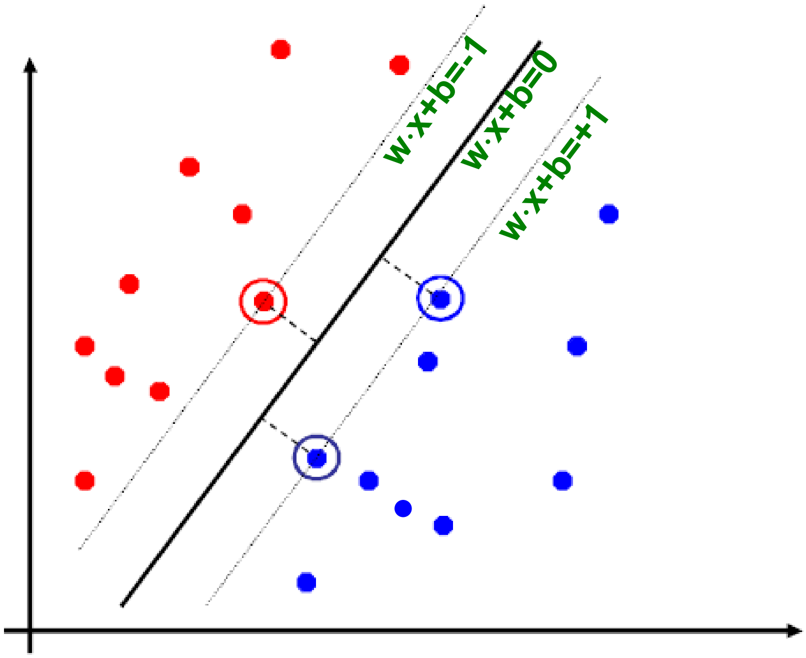
and since any support vector has y = wx + b,

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

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

SVM: Support Vectors

- Two ways to store an SVM model in memory:
  1. Store the parameter values for w, b
  2. Store $\left(x^{(i)}, y^{(i)}\right), \alpha^{(i)}$ for all support vectors i



SVM with soft margin

$$\min_{w,b,\xi_i \geq 0} \; \frac{1}{2}\|w\|^2 + R \cdot \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i$$

Dual form

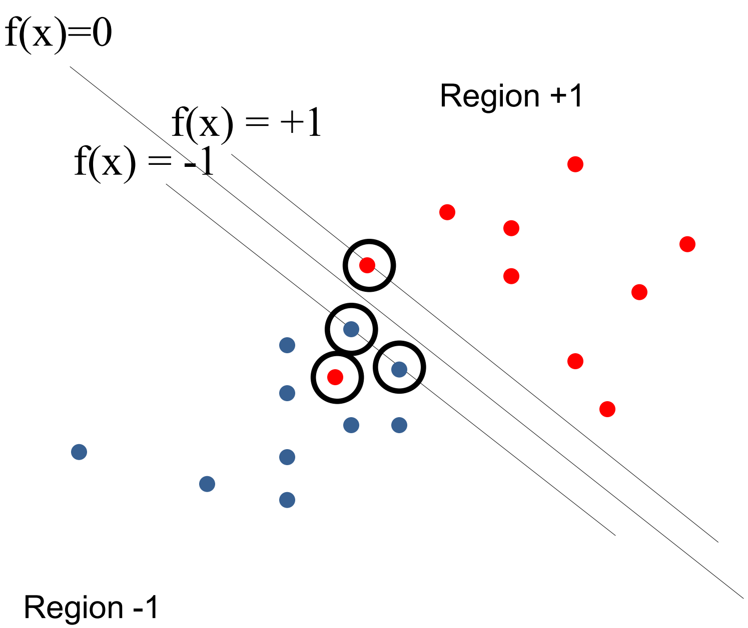
- Soft margin dual:

K_{ij}

$$\max_{0 \le \alpha \le R} \sum_i \alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j \, y^{(i)} y^{(j)} \, (x^{(i)} \cdot x^{(j)})$$

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…

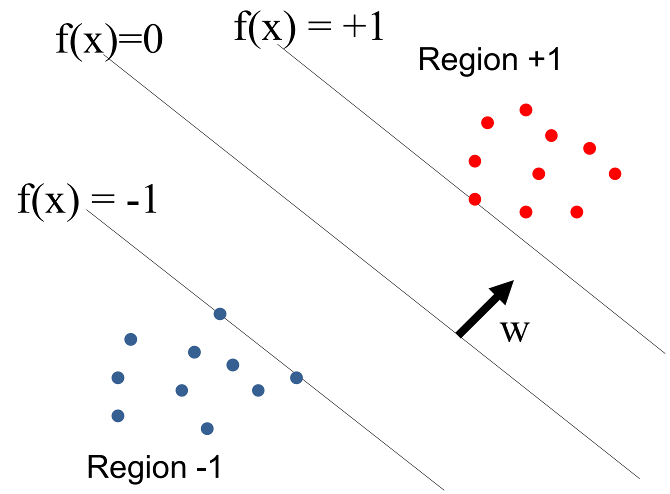
Prediction:

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

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

$$b = \ldots$$

Linear SVMs



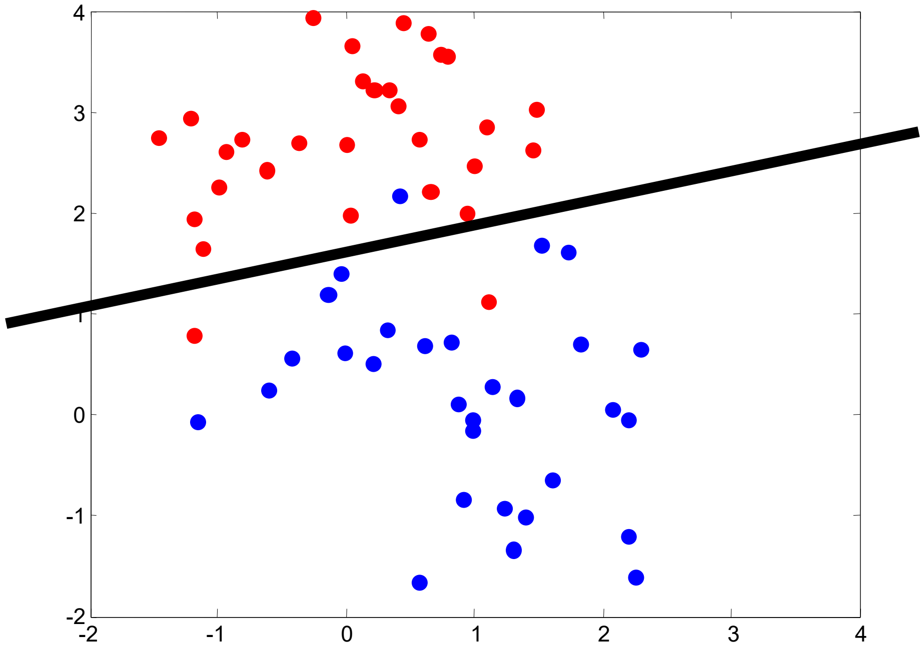
- So far, looked at linear SVMs:
  - Expressible as linear weights "w"
  - Linear decision boundary

- Dual optimization for a linear SVM:

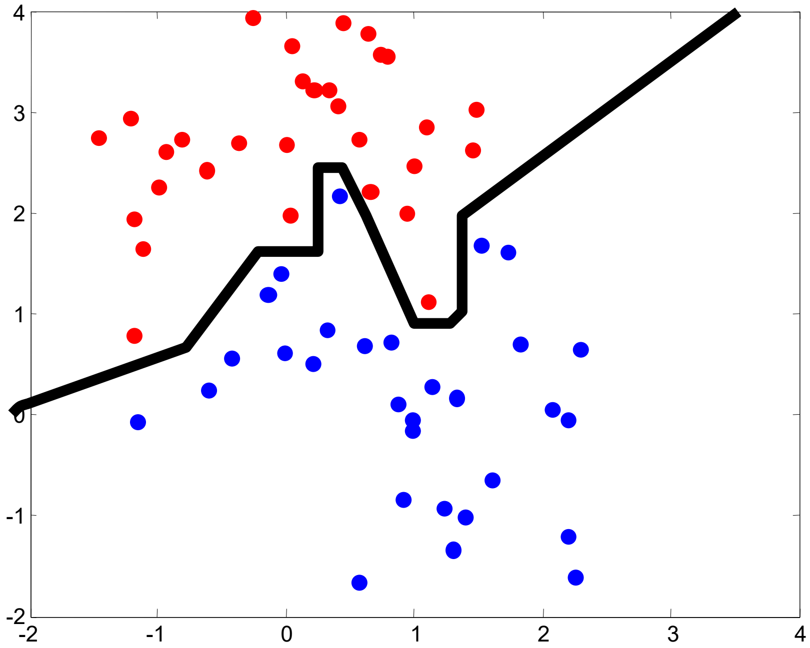
$$\max_{0 \le \alpha \le R} \sum_i \alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j \, y^{(i)} y^{(j)} \left(x^{(i)} \cdot x^{(j)} \right) \qquad \text{s.t.} \; \sum_i \alpha_i y^{(i)} = 0$$

- Depend on pairwise dot products:
  - Kij measures "similarity", e.g., 0 if orthogonal $K_{ij} = x^{(i)} \cdot x^{(j)}$

linear decision boundary



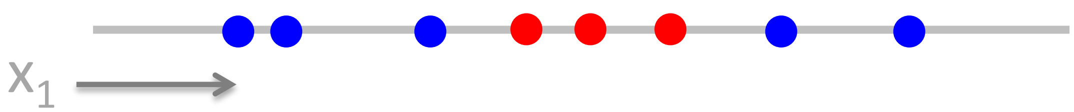
Non-linear decision boundary



Adding features

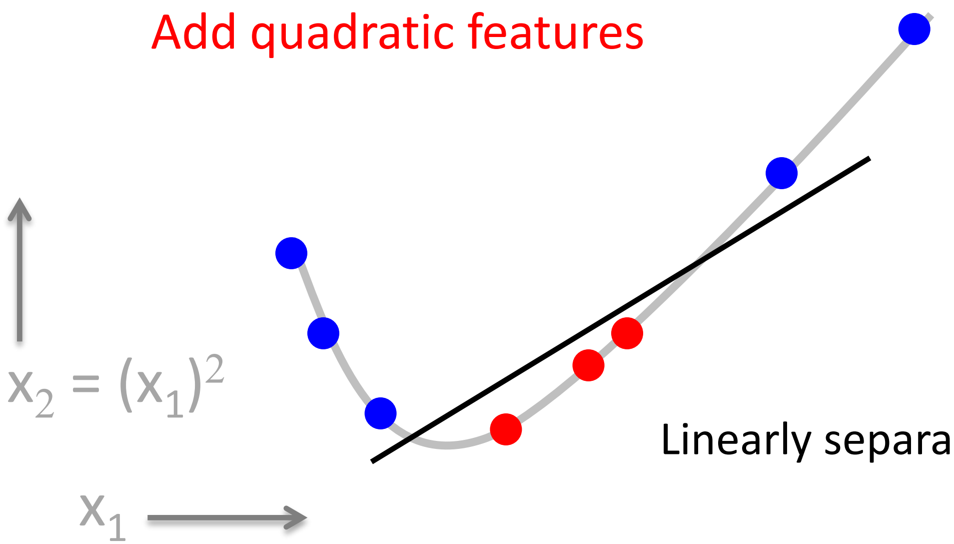
- Linear classifier can't learn some functions

1D example:



Not linearly separable

Add quadratic features

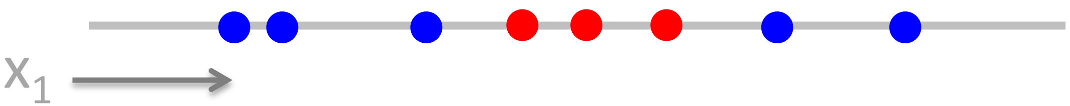


Linearly separable in new features…

Adding features

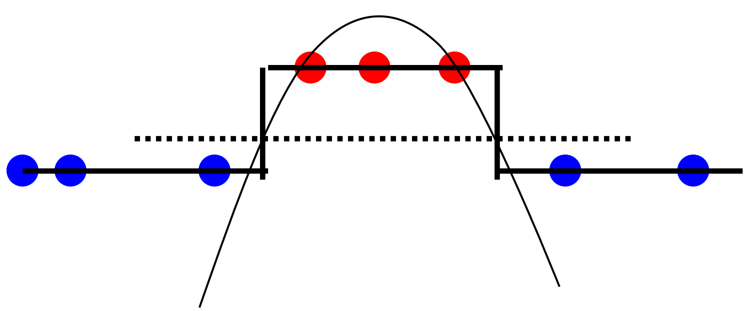
- Linear classifier can’t learn some functions

1D example:



Not linearly separable

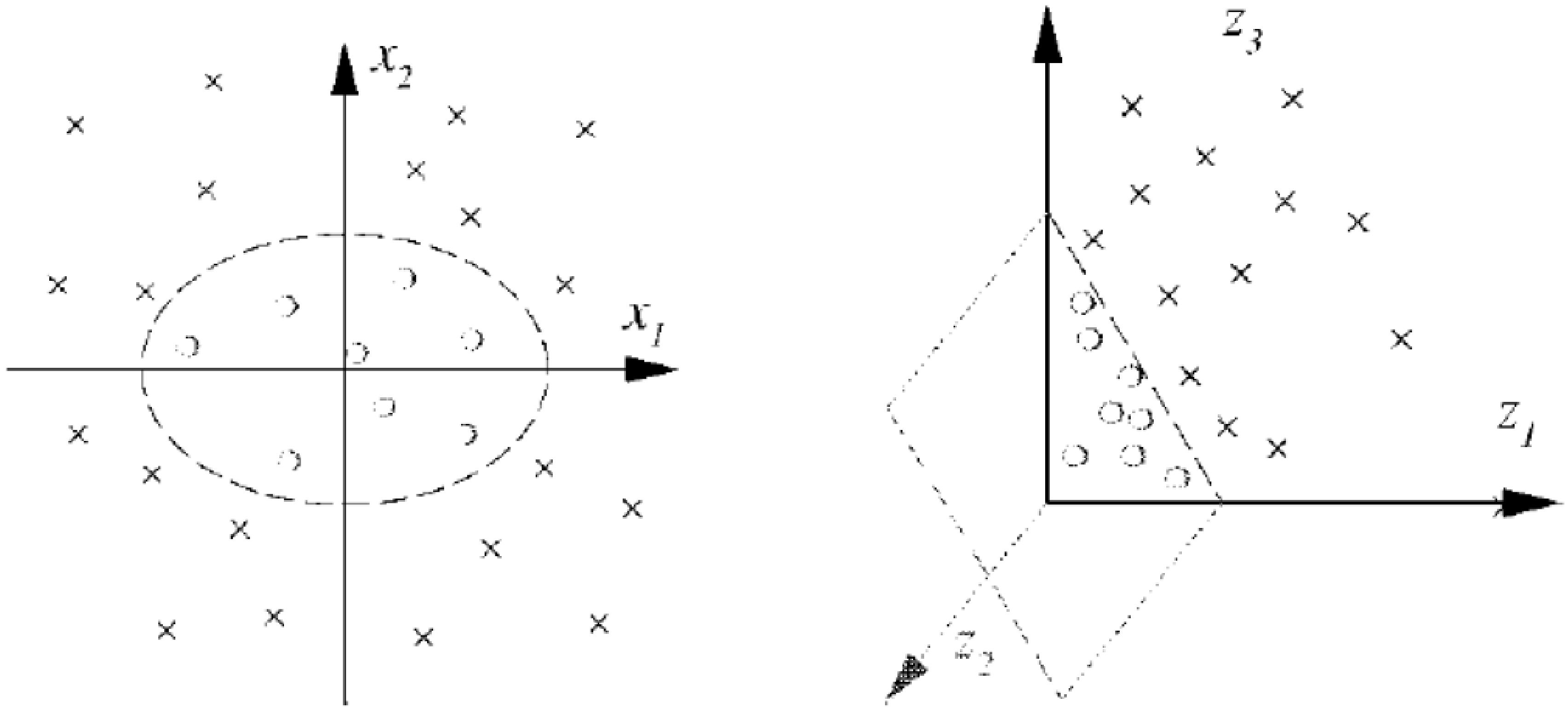
Quadratic features, visualized in original feature space:



$$y = T(\, a\, x^2 + b\, x + c \,)$$

More complex decision boundary: $ax^2+bx+c = 0$

Example



$$z_1 = x_1^2, \quad z_2 = \sqrt{2}x_1x_2, \quad z_3 = x_2^2$$

Adding features

- Feature function Phi(x)
  - Predict using some transformation of original features

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

- Dual form of SVM optimization is:

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

- For example, quadratic (polynomial) features:

$$\Phi(x) = \big(1 \;\; \sqrt{2} x_1 \;\; \sqrt{2} x_2 \;\; \cdots \;\; x_1^2 \;\; x_2^2 \;\; \cdots \;\; \sqrt{2} x_1 x_2 \;\; \sqrt{2} x_1 x_3 \;\; \cdots \big)$$

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

Implicit features

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

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

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

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

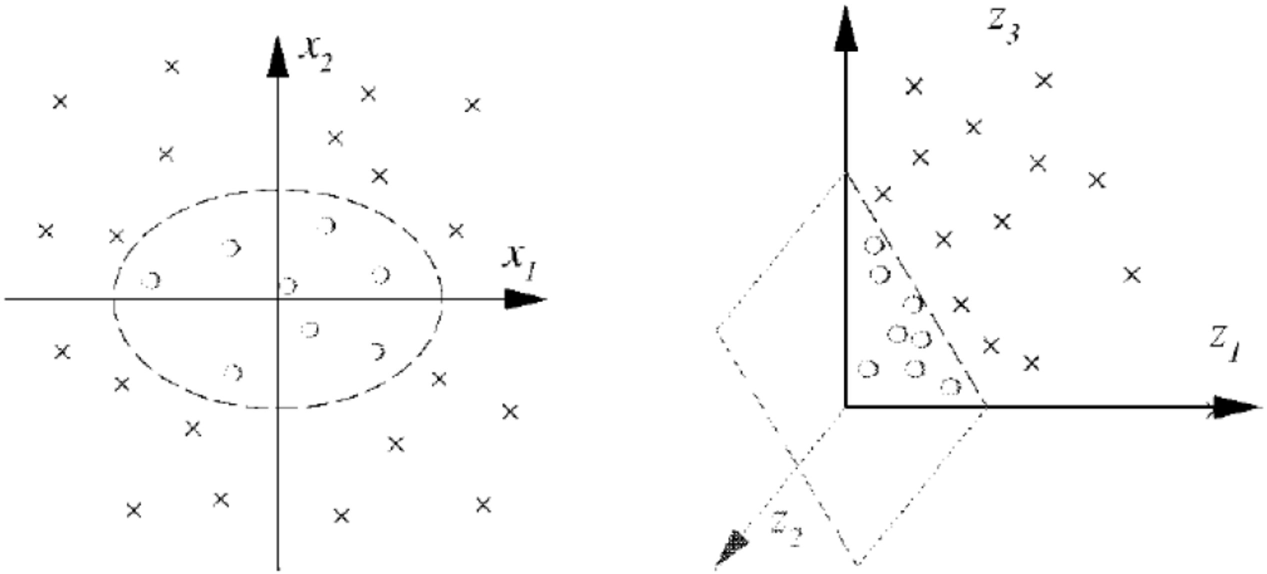
$$\Phi(a)^T\Phi(b) = 1 + \sum_j 2a_jb_j + \sum_j a_j^2b_j^2 + \sum_j \sum_{k>j} 2a_ja_kb_jb_k + \ldots$$

$$= (1 + \sum_j a_jb_j)^2 \quad = (1 + a \cdot b)^2$$

$$= K(a, b)$$

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

Example



$$z_1 = x_1^2, \quad z_2 = \sqrt{2}x_1x_2, \quad z_3 = x_2^2$$

$$\begin{aligned}\Phi(a)\Phi(b) &= \left(a_1^2, \sqrt{2}a_1a_2, a_2^2\right)\cdot\left(b_1^2, \sqrt{2}b_1b_2, b_2^2\right)\\ &= (a_1^2b_1^2 + 2a_1b_1a_2b_2 + a_2^2b_2^2)\\ &= (a_1b_1 + a_2b_2)^2 = (a\cdot b)^2 = K(a,b)\end{aligned}$$

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:

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

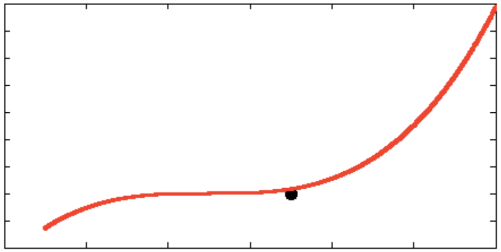
- Then, $K(a,b) = \Phi(a) \cdot \Phi(b)$ for some $\Phi(x)$

- Notably, Phi 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

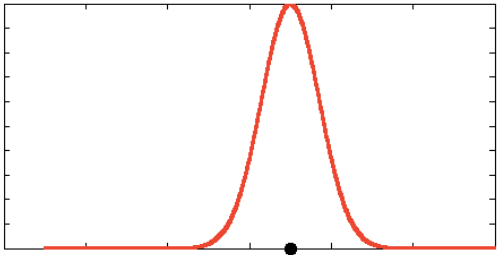
- Some commonly used kernel functions & their shape:

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



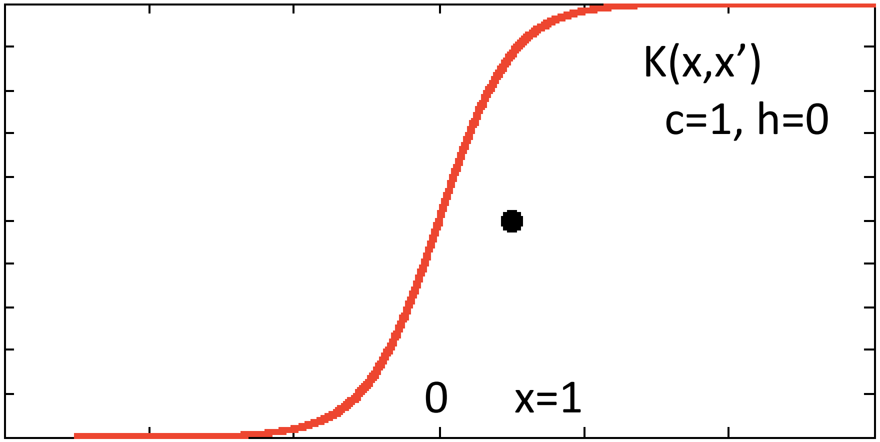
- Radial Basis Functions

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



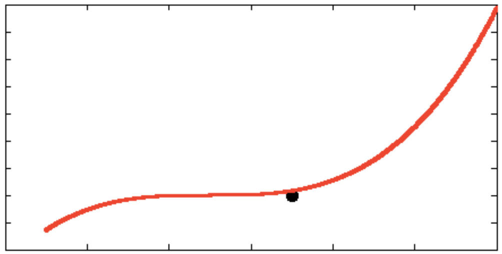
- Saturating, sigmoid-like:

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



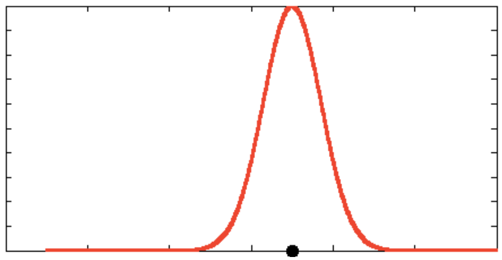
Common kernel functions

- Some commonly used kernel functions & their shape:



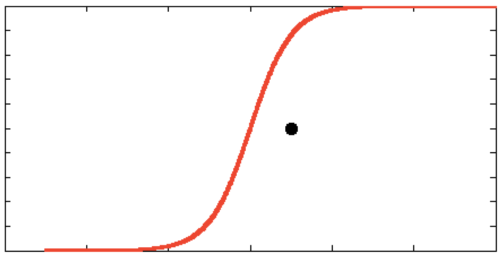
- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$
- Radial Basis Functions

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



- 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…

Kernel SVMs

- Learning:

$$\arg\max_{0 \le \alpha_i \le R} \sum_i \alpha_i - \frac{1}{2} \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(x^{(i)}, x^{(j)}\right)$$

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

- Prediction:

$$\hat{y} = \sum_{i:\alpha_i > 0} \alpha_i y^{(i)} K\left(x^{(i)}, x\right) + b$$

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
  - Maximize margin subject to linear constraints
- "Soft margin" for non-separable data
  - Regularized hinge loss
- Kernels
  - Dual form involves only pairwise similarity
  - Mercer kernels: dot products in implicit high-dimensional space