CS6510
Applied Machine Learning

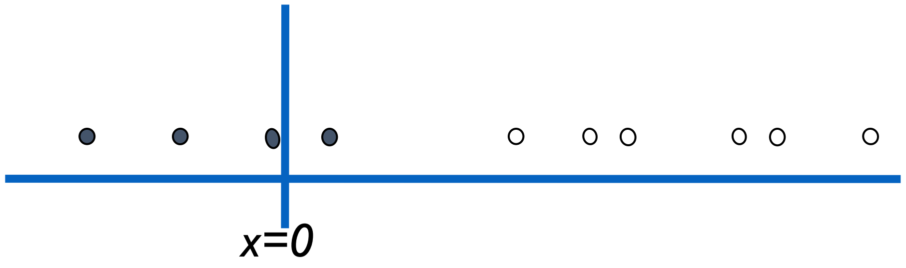
Kernel Classifiers

Slide credits : Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

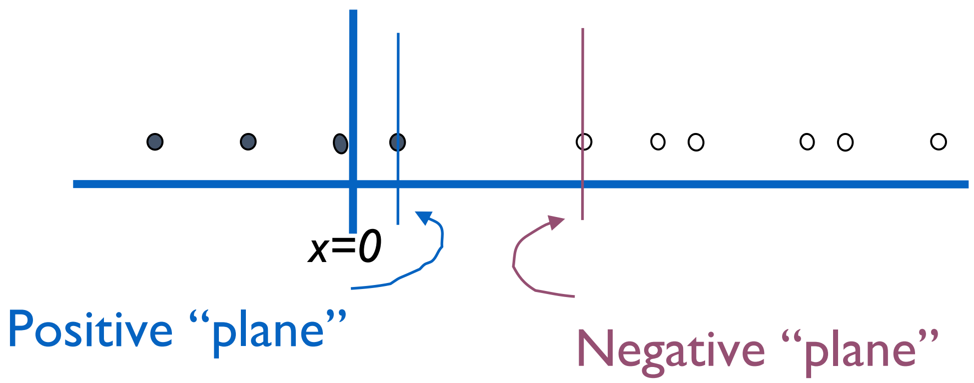
Assume we are in 1-dimension

What would SVMs do with this data?



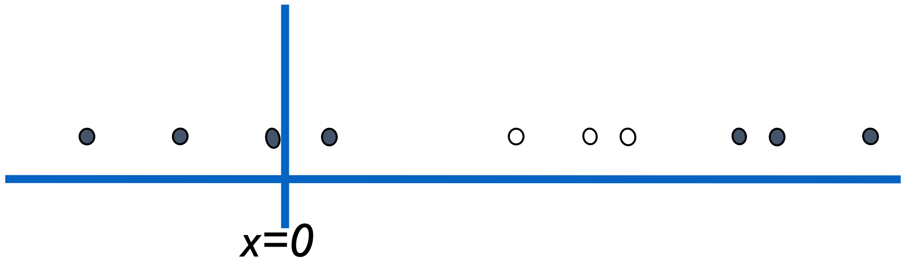
Assume we are in 1-dimension

Not a big surprise

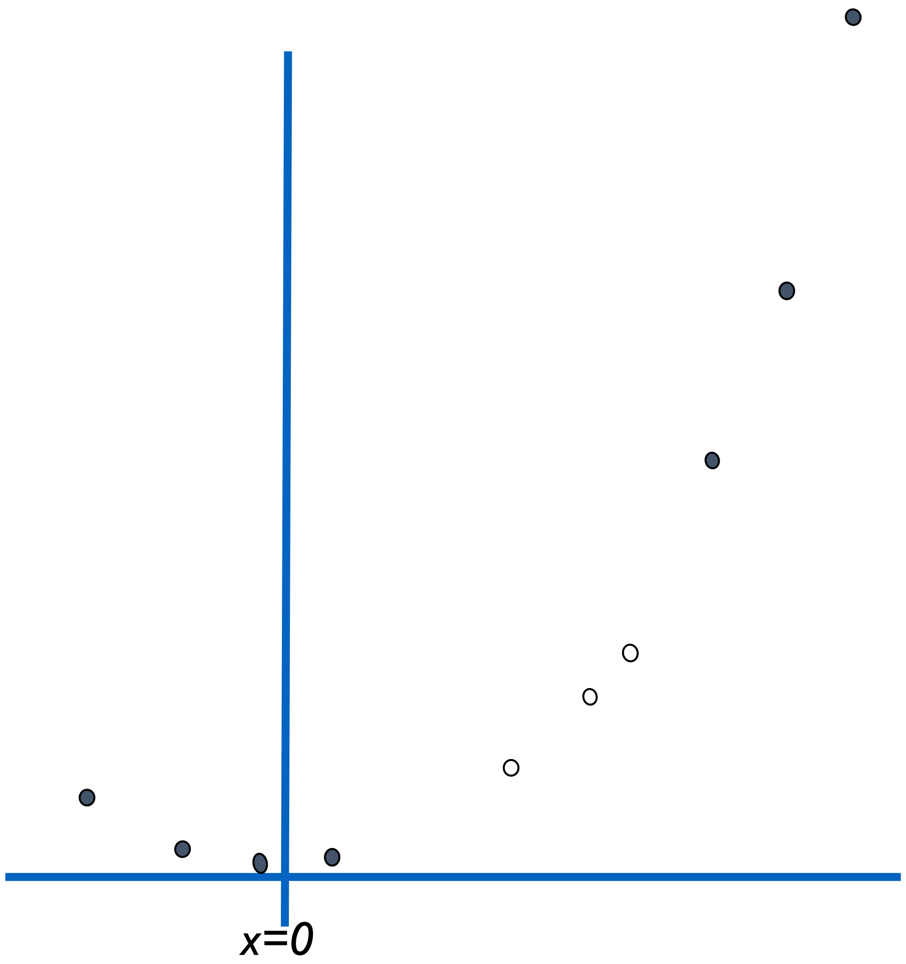


Harder 1-dimensional Dataset

What can be done about this?



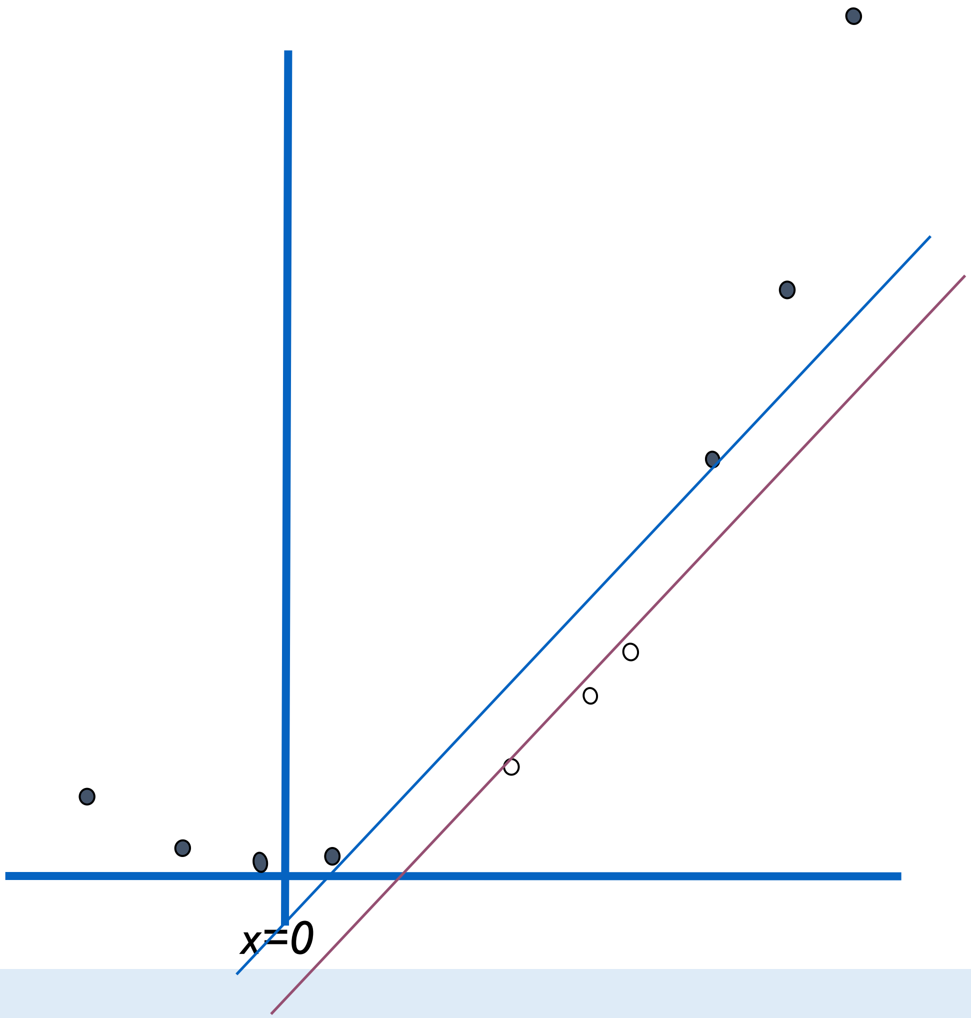
Harder 1-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, x_k^2)$$

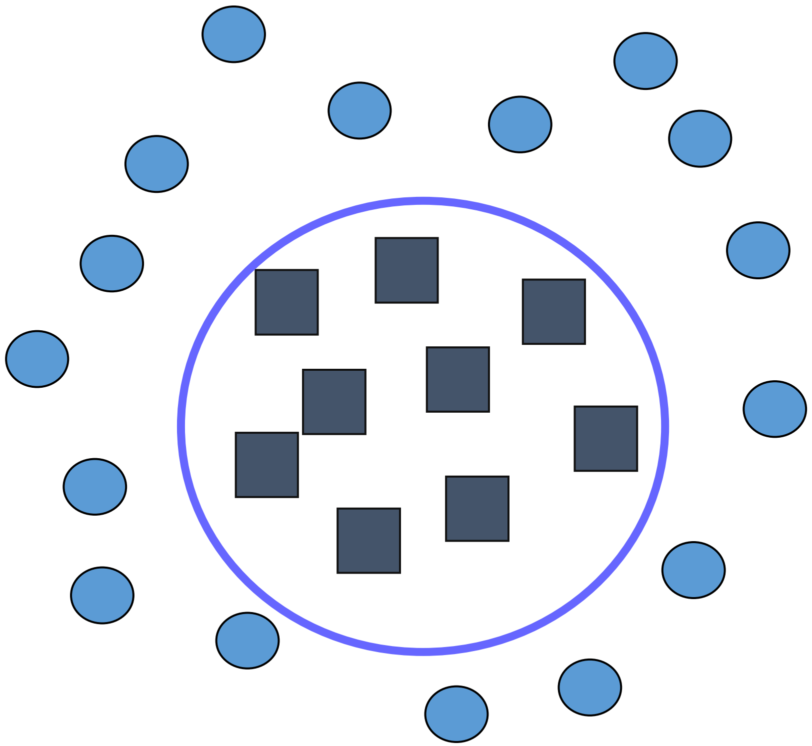
Harder 1-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, x_k^2)$$

Harder 2-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, y_k, x_k^2, y_k^2, x_k y_k)$$

Other Mapping Functions

$\mathbf{z}_k$ = (polynomial terms of $\boldsymbol{x}_k$ of degree 1 to q)

$\mathbf{z}_k$ = (radial basis functions of $\boldsymbol{x}_k$)

$$\mathbf{z}_k[j] = \varphi_j(\mathbf{x}_k) = \exp\left(-\frac{|\mathbf{x}_k - \mathbf{c}_j|^2}{\sigma^2} \right)$$

$\mathbf{z}_k$ = (sigmoid functions of $\boldsymbol{x}_k$)

Recall: SVM Lagrangian Dual

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

subject to constraints: $0 \le \alpha_k \le c \quad \forall k \qquad \sum_{k=1}^{R} \alpha_k y_k = 0$

Once solved, we obtain w and b using:

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

$$y_i (x_i \bullet w + b) - 1 = 0$$

$$b = -y_i (y_i (x_i \bullet w) - 1)$$

Then classify with:

***f*(x,w,*b*)** = *sign*(**w. x** + *b*)

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

subject to constraints: $0 \le \alpha_k \le C \quad \forall k \qquad \sum_{k=1}^{R} \alpha_k y_k = 0$

Then compute:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

Then classify with:

f*(*x*,*w*,*b*)** = *sign*(w*.** $\mathbf{\Phi}$(***x***) + *b*)

Most important change:

x → $\mathbf{\Phi}$ (***x***)

Quadratic Dot Products

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}$$

$$1 + \sum_{i=1}^{m} 2a_ib_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

Number of terms (assuming m input dimensions) = (m+2)-choose-2 = (m+2)(m+1)/2 = (approx) $m^2/2$

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

subject to constraints: $0 \le \alpha_k \le C$

Then compute:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

We must do $R^2/2$ dot products to get this matrix ready

Assuming a quadratic polynomial kernel, each dot product requires $m^2/2$ additions and multiplications (where m is the dimension of x)

The whole thing costs $R^2 m^2 /4$.

Quadratic Dot Products

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

Quadratic Dot Products

They're the same!
And this is only O(m) to compute!

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

आई आई टी हैदराबाद
IIT Hyderabad

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\boldsymbol{\Phi}(\mathbf{x}_k) . \boldsymbol{\Phi}(\mathbf{x}_l))$

subject to constraints: $0 \le \alpha_k \le C$

We must do $R^2/2$ dot products to get this matrix ready

Now, each dot product now only requires m additions and multiplications

Then compute:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \boldsymbol{\Phi}(\mathbf{x}_k)$$

Most important change:

$\boldsymbol{x} \rightarrow \boldsymbol{\Phi}(\boldsymbol{x})$

Higher-Order Polynomials

Poly-nomial	f(x)	Cost to build Q_{kl} matrix traditionally	Cost if 100 dimensions	f(a).f(b)	Cost to build Q_{kl} matrix sneakily	Cost if 100 dimensions
Quadratic	All $m^2/2$ terms up to degree 2	$m^2 R^2 /4$	2,500 R^2	$(a.b+1)^2$	$m R^2 / 2$	50 R^2
Cubic	All $m^3/6$ terms up to degree 3	$m^3 R^2 /12$	83,000 R^2	$(a.b+1)^3$	$m R^2 / 2$	50 R^2
Quartic	All $m^4/24$ terms up to degree 4	$m^4 R^2 /48$	1,960,000 R^2	$(a.b+1)^4$	$m R^2 / 2$	50 R^2

आई आई टी हैदराबाद
IIT Hyderabad

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l K(\mathbf{x}_k, \mathbf{x}_l)$

Kernel gram matrix

Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\qquad \sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

Most important change:

$$\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l) \rightarrow K(\mathbf{x}_k, \mathbf{x}_l)$$

$$y(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^\top \phi(\mathbf{x})) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})^\top \phi(\mathbf{x})\right)$$
$$= \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x})\right).$$

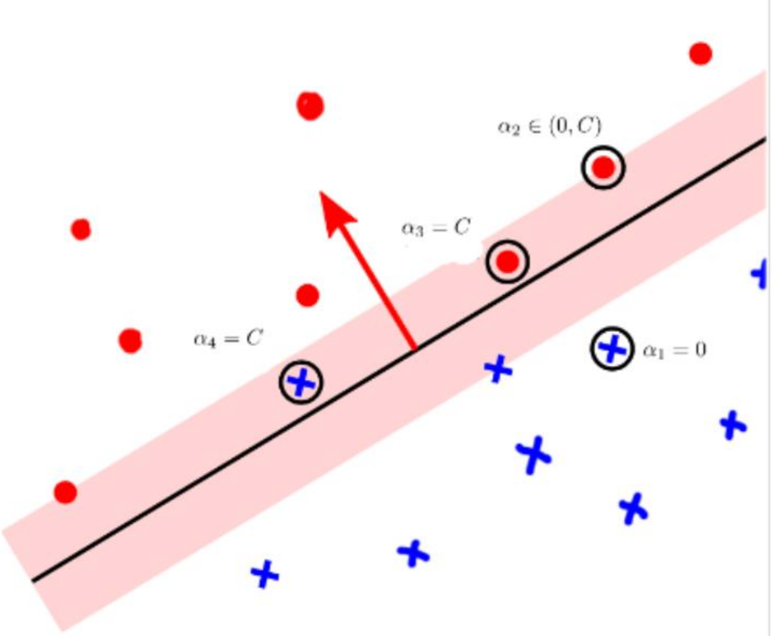
SVMs and Dual variables

There are 3 kinds of data vectors $\mathbf{x}_n$.

1. Non-support vectors. Examples that lie on the correct side outside the margin, so $\alpha_n = 0$.

2. Essential support vectors. Examples that lie just on the margin, therefore $\alpha_n \in (0, C)$.

3. Bound support vectors. Examples that lie strictly inside the margin, or on the wrong side, therefore $\alpha_n = C$



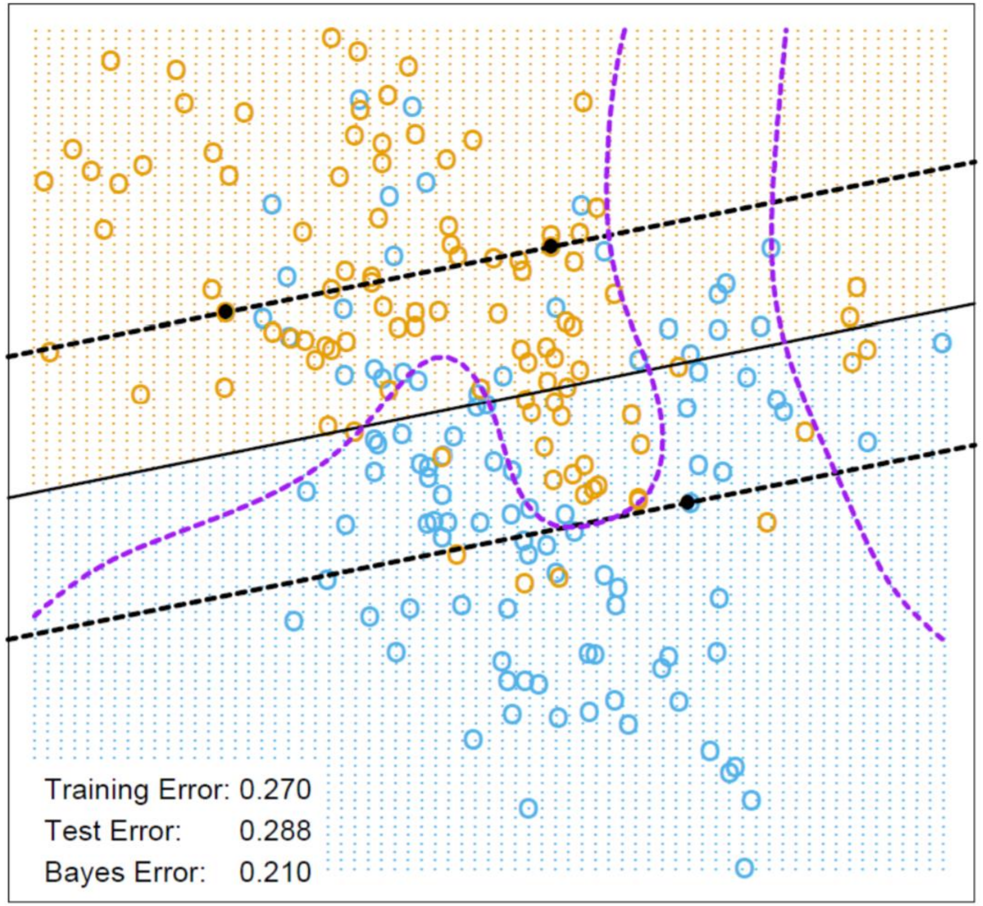
SVM Kernel Functions

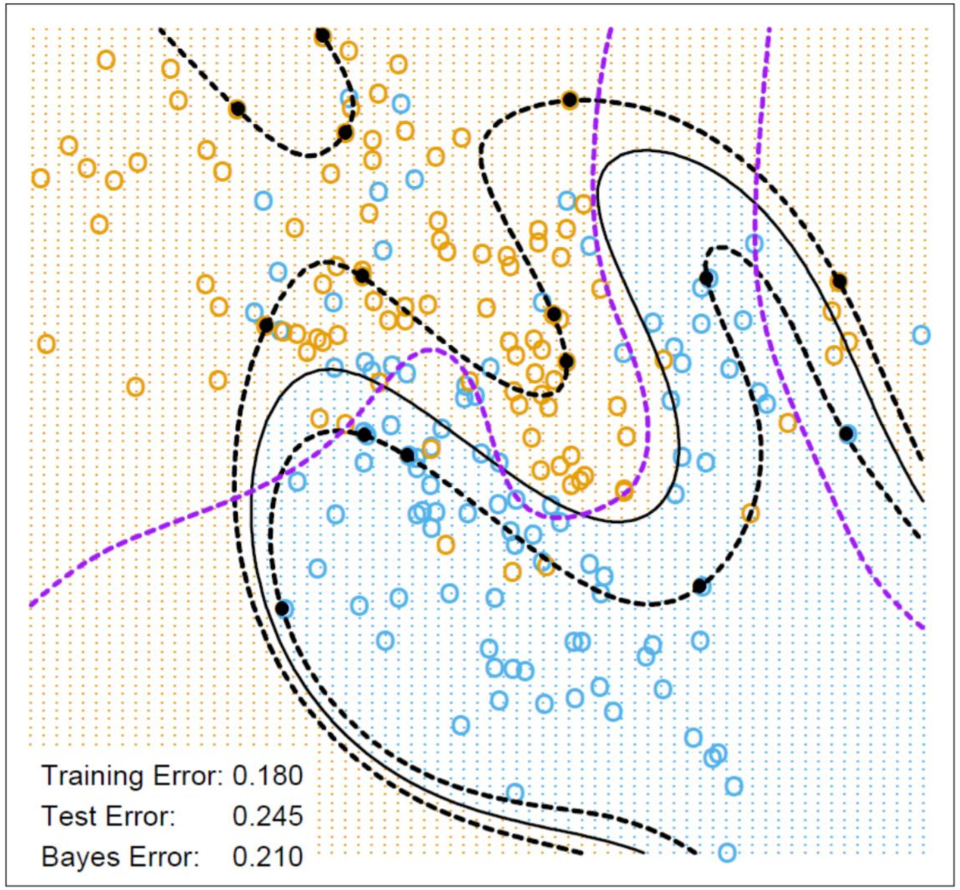
- $K(\boldsymbol{a},\boldsymbol{b})=(\boldsymbol{a}\,.\,\boldsymbol{b}+1)^d$ is an example of a **kernel function** in SVM
- Beyond polynomials, there are other high-dimensional kernel functions such as:
  - Radial-Basis-style Kernel Function:

$$K(\mathbf{a},\mathbf{b}) = \exp\left(-\frac{(\mathbf{a}-\mathbf{b})^2}{2\sigma^2}\right)$$

  - Sigmoidal function

Linear vs non linear





Kernel Tricks

- Replacing dot product with a kernel function
- Not all functions are kernel functions
  - Need to be decomposable: $K(a,b) = \phi(a) \cdot \phi(b)$
- **Mercer's condition** To expand Kernel function $K(x,y)$ into a dot product, i.e. $K(x,y)=\Phi(x)\cdot\Phi(y)$, $K(x, y)$ has to be positive semi-definite function, i.e., for any function $f(x)$ whose $\int f^2(x)dx$ is finite, the following inequality holds:

$$\int dxdy f(x)K(x,y)f(y) \geq 0$$

- Positive constant function is a kernel: for $\alpha \geq 0$, $K'(x_1, x_2) = \alpha$
- Positively weighted linear combinations of kernels are kernels: if $\forall i, \alpha_i \geq 0$, $K'(x_1, x_2) = \sum_i \alpha_i K_i(x_1, x_2)$
- Products of kernels are kernels: $K'(x_1, x_2) = K_1(x_1, x_2) K_2(x_1, x_2)$
- The above transformations preserve positive semidefinite functions

How to choose a kernel function?

- Not easy! Remember – this depends on your data geometry
- If linear works, go with it
- RBF kernels are considered good in general, especially for images (and other smooth functions/data)
- For discrete data, chi-square kernel preferred of late (especially for histogram data)
- You can also do Multiple Kernel Learning
- Still not sure? Use cross-validation to select a kernel function from some basic options

An excellent resource: http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

Kernelizing other Methods

- The same kernel trick can also be applied to other methods including:
  - Kernel k-NN
  - Kernel Perceptron (we will see later)
  - Kernelized Linear Regression (we will see later)

Representer Theorem

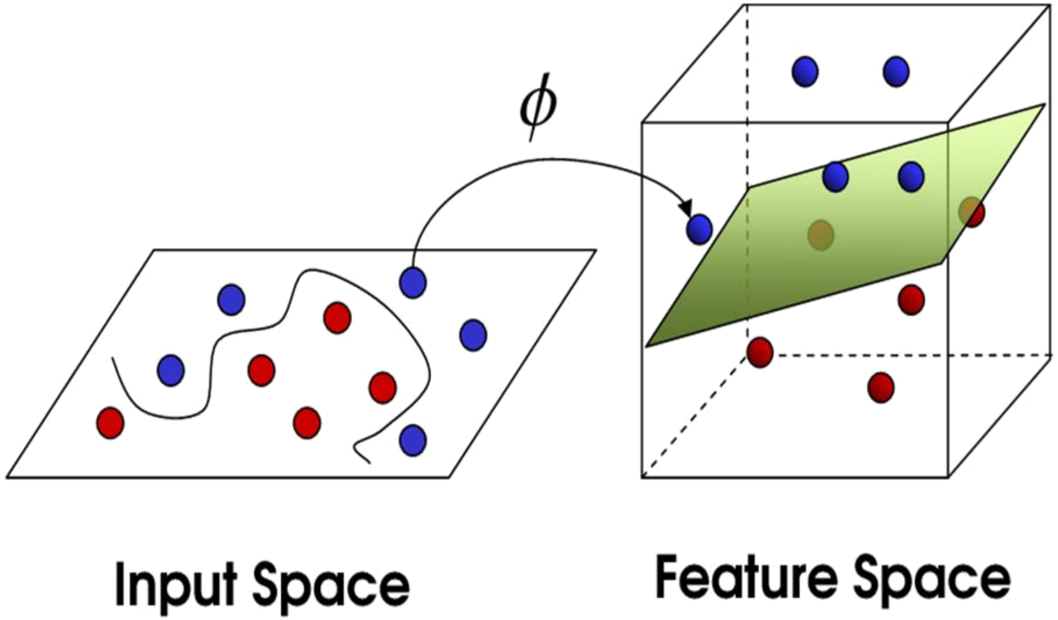
If $\mathbf{w}^*$ is defined as

$$\mathbf{w}^* = \arg\min \sum_{i=1}^{N} L\left(\left\langle \mathbf{w}, \phi(\mathbf{x}^{(i)}) \right\rangle, t^{(i)}\right) + \lambda \|\mathbf{w}\|^2,$$

then $\mathbf{w}^* \in \mathrm{span}\{\phi(x_1), ..., \phi(x_N)\}$, i.e. $\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i \phi(x_i)$ for some $\alpha \in \mathbb{R}^N$.

Non-Linear Regression

- Recall: “kernel trick”
- **Key Idea:** Map data to higher dimensional space (feature space) and perform linear regression in embedded space



Ridge Regression

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}.$$

- Regularized Least Squares

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

N × M

- Show that the regularized least squares solution is

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

Stable and unique solution

Regression Methods

- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression
- Logistic Regression

Readings

- PRML, Bishop, Chapter 7 (7.1-7.3)
- "Introduction to Machine Learning" by Ethem Alpaydin, 2nd edition, Chapters 3 (3.1-3.4), Chapter 13 (13.1-13.9)
- For kernel functions:
  - http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/