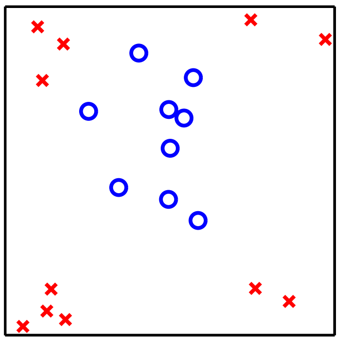
Learning From Data
Lecture 26
Kernel Machines

Popular Kernels
The Kernel Measures Similarity
Kernels in Different Applications

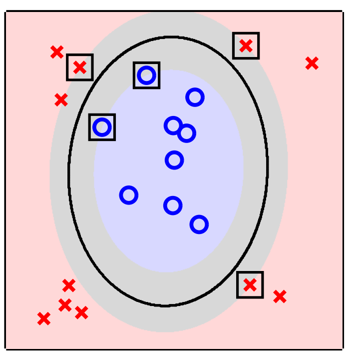
M. Magdon-Ismail
CSCI 4100/6100

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



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

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



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

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

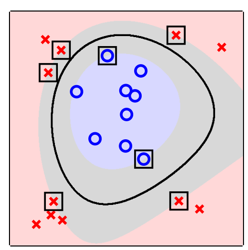
(One can compute b^* for several SVs and average)

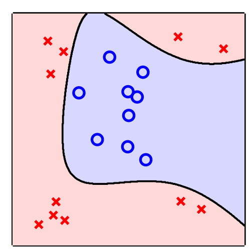
Solve the QP

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

Overfitting

SVM | Pseudo-inverse





high $\tilde{d} \to$ complicated separator

few support vectors $\to$ low effective complexity

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

Computation

Inner products with Kernel

$K(\cdot, \cdot)$

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

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

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

Polynomial Kernel

2nd-Order Polynomial Kernel

$$\Phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_d^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_d \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_{d-1}x_d \end{bmatrix}$$

$$\begin{aligned} K(\mathbf{x},\mathbf{x}') &= \Phi(\mathbf{x})^{\mathrm{T}}\Phi(\mathbf{x}') \\ &= \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d} x_i^2 x_i'^2 + 2\sum_{i<j} x_i x_j x_i' x_j' && \leftarrow O(d^2) \\ &= \left(\frac{1}{2} + \mathbf{x}^{\mathrm{T}}\mathbf{x}'\right)^2 - \frac{1}{4} \end{aligned}$$

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

Q-th order polynomial kernel

$$K(\mathbf{x},\mathbf{x}') = (r + \mathbf{x}^{\mathrm{T}}\mathbf{x}')^{Q} \qquad \longleftarrow \text{ inhomogeneous kernel}$$

$$K(\mathbf{x},\mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}')^{Q} \qquad \longleftarrow \text{ homogeneous kernel}$$

RBF-Kernel

One dimensional RBF-Kernel

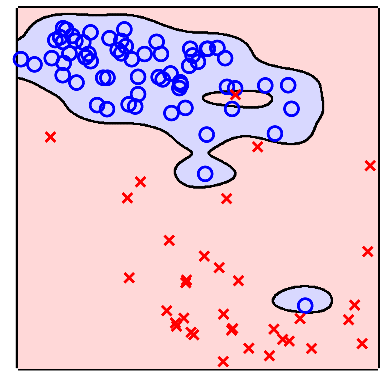
$$\Phi(x) = e^{-x^2}\begin{bmatrix} 1 \\ \sqrt{\frac{2^1}{1!}}x \\ \sqrt{\frac{2^2}{2!}}x^2 \\ \sqrt{\frac{2^3}{3!}}x^3 \\ \sqrt{\frac{2^4}{4!}}x^4 \\ \vdots \end{bmatrix}$$

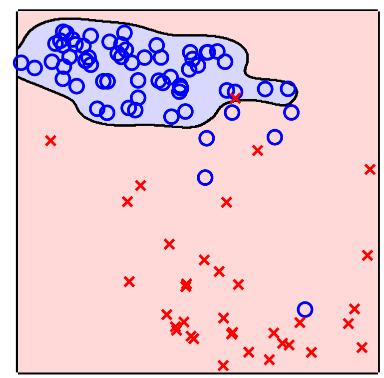
$$\begin{aligned} K(\mathbf{x},\mathbf{x}') &= \Phi(\mathbf{x})^{\mathrm{T}}\Phi(\mathbf{x}') \\ &= e^{-x^2}e^{-x'^2}\sum_{i=0}^{\infty}\frac{(2xx')^i}{i!} \qquad \leftarrow \text{not feasible} \\ &= e^{-x^2}e^{-x'^2}e^{2xx'} \\ &= e^{-(x-x')^2} \end{aligned}$$

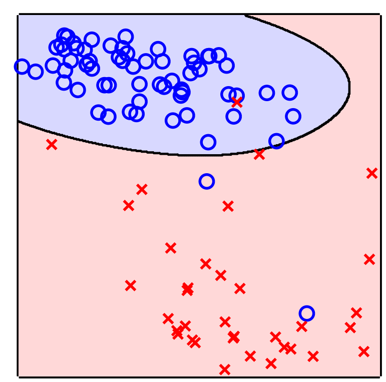
$\uparrow$
computed quickly
in $\mathcal{X}$-space, in $O(d)$

d-dimensional RBF-Kernel

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

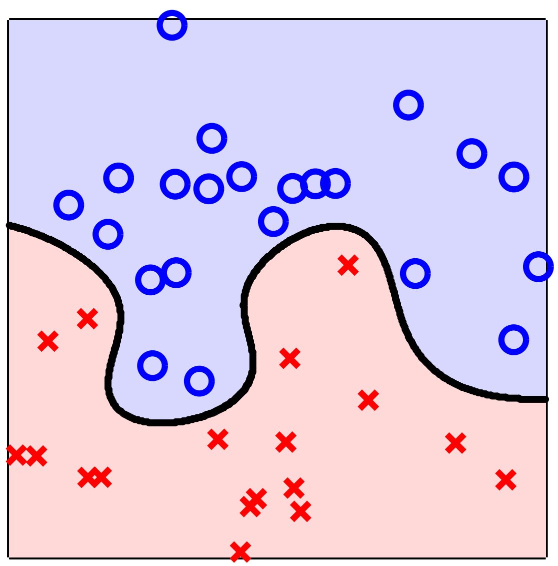

Hard Margin ($\gamma = 2000,\ C = \infty$)


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

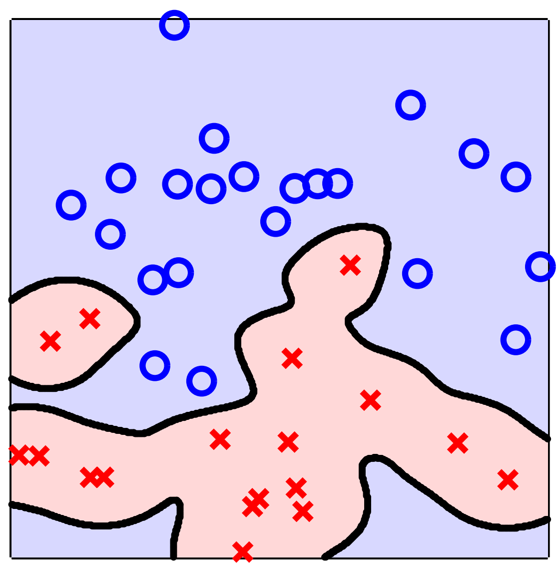

Soft Margin ($\gamma = 100,\ C = 0.25$)

Choosing RBF-Kernel Width γ

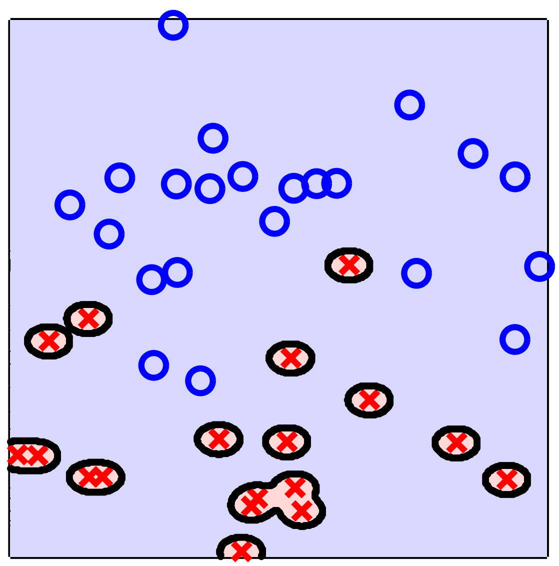
$$e^{-\gamma \| \mathbf{x} - \mathbf{x}' \|^2}$$



Small γ



Medium γ



Large γ!

RBF-Kernel Simulates k-RBF-Network

<u>RBF-Kernel</u>

$$g(\mathbf{x}) = \text{sign}\left(\sum_{\alpha_n^*>0} \alpha_n^* y_n e^{-\|\mathbf{x}-\mathbf{x}_n\|^2} + b^*\right)$$

Centers are at support vectors
Number of centers auto-determined

<u>k-RBF-Network</u>

$$g(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{k} w_j e^{-\|\mathbf{x}-\boldsymbol{\mu}_j\|^2} + w_0\right)$$

Centers chosen to represent the data
Number of centers k is an input

Neural Network Kernel

$$K(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \cdot \mathbf{x}^{\mathrm{T}} \mathbf{x}' + c)$$

<u>Neural Network Kernel</u>

$$g(\mathbf{x}) = \text{sign}\left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n \tanh(\kappa \cdot \mathbf{x}_n{}^{\mathrm{T}} \mathbf{x} + c) + b^*\right)$$

First layer weights are support vectors
Number of hidden nodes auto-determined

<u>2 Layer Neural Network</u>

$$g(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{m} w_j \tanh(\mathbf{v}_j{}^{\mathrm{T}} \mathbf{x}) + w_0\right)$$

First layer weights arbitrary
Number of hidden nodes m is an input

The Inner Product Measures Similarity

$$K(\mathbf{x},\mathbf{x}') = \mathbf{z}^{\mathrm{T}}\mathbf{z}' \;=\; \|\,\mathbf{z}\,\| \cdot \|\,\mathbf{z}'\,\| \cdot \cos(\theta_{\mathbf{z},\mathbf{z}'})$$

$$= \|\,\mathbf{z}\,\| \cdot \|\,\mathbf{z}'\,\| \cdot CosSim(\mathbf{z},\mathbf{z}')$$

Normalizing for size, Kernel measures similarity between input vectors

Designing Kernels

- Construct a similarity measure for the data

- A linear model should be plausible in that transformed space

String Kernels

Applications: DNA sequences, Text

ACGGTGTCAAACGTGTCAGTGTG

GTCGGGTCAAAACGTGAT

```
Dear Sir,
With reference to your letter dated 26th
March, I want to confirm the Order No.
34-09-10 placed on 3rd March, 2010.  I
would appreciate if you could send me
the account details where the payment has
to be made.  As per the invoice, we are
entitled to a cash discount of 2%.  Can
you please let us know whether it suits
you if we make a wire transfer instead of
a cheque?
```

```
Dear Jane,
I am terribly sorry to hear the news of
your hip fracture.  I can only imagine
what a terrible time you must be going
through.  I hope you and the family are
coping well.  If there is any help you
need, don't hesitate to let me know.
```

Similar?

Yes, if classifying spam versus non-spam
No, if classifying business versus personal

To design the kernel $\longrightarrow$ measure similarity between strings

Bag of words (number of occurences of each atom)

Co-occurrence of substrings or subsequences

Graph Kernels

Performing classification on:

Graph structures (eg. protein networks for function prediction)

Graph nodes within a network (eg. advertise of not to Facebook users)

Similarity between **graphs**:

random walks degree sequences, connectivity properties, mixing properties.

Measuring similarity between **nodes**:

Looking at neighborhoods, $K(v, v') = \dfrac{|N(v) \cap N(v')|}{|N(v) \cup N(v')|}$.

Image Kernels

Similar?

Yes - if trying to regcognize pictures with faces.

No - if trying to distinguish Malik from Christos