Kernels

Dr. Fayyaz ul Amir Afsar Minhas
PIEAS Biomedical Informatics Research Lab
Department of Computer and Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad, Pakistan
http://faculty.pieas.edu.pk/fayyaz/

Transformations

- Transformations can be used to make the data linearly separable
- But it may not always be possible to find a transformation
  - Use a soft-SVM

Another look at the SVM

$$\max_{\alpha_i,\beta_i\geq 0}\left\{\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}^{(i)^T}\mathbf{x}^{(j)}\right\}$$

$$s.t \qquad 0\leq\alpha_i\leq C, \qquad \sum_{i=1}^{N}\alpha_i y_i=0$$

- We can replace the dot product $\boldsymbol{x}^{(i)^T}\boldsymbol{x}^{(j)}$ with:
  - a generalized dot product (inner product)
    - $\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\rangle = \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^T\boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right)$
    - Advantage: can implement feature transformations
  - or a function (called the kernel function)
    - $\boldsymbol{K_{ij}} = \boldsymbol{K}(\boldsymbol{i},\boldsymbol{j})$
    - Advantage: No need for explicit feature representation

From Transforms to Kernels

- For the XOR problem we defined the transformation
- $\boldsymbol{\phi}\left(\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\end{bmatrix}\right) = \begin{bmatrix}(\boldsymbol{x_1})^2\\ (\boldsymbol{x}_2)^2\\ \sqrt{2}\boldsymbol{x_1}\boldsymbol{x_2}\end{bmatrix}$
- Let's compute the dot product of two examples in the transformed space
- $\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\rangle = \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^{\boldsymbol{T}}\boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right) = \left[\left(x_1^{(i)}\right)^2 \quad \left(x_2^{(i)}\right)^2 \quad \sqrt{2}x_1^{(i)}x_2^{(i)}\right]\begin{bmatrix}\left(x_1^{(j)}\right)^2\\ \left(x_2^{(j)}\right)^2\\ \sqrt{2}x_1^{(j)}x_2^{(j)}\end{bmatrix} =$
- $\left(x_1^{(i)}x_1^{(j)}\right)^2 + \left(x_2^{(i)}x_2^{(j)}\right)^2 + 2x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)} = \left(x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)}\right)^2 = \left(\left(\boldsymbol{x}^{(i)}\right)^{\boldsymbol{T}}\left(\boldsymbol{x}^{(j)}\right)\right)^2$
- The inner product (dot product) in the transform space is the square of the dot product of the original space
- This is the polynomial kernel of degree 2
- $k(i,j) = \left(\left(\boldsymbol{x}^{(i)}\right)^{\boldsymbol{T}}\left(\boldsymbol{x}^{(j)}\right)\right)^2$

- What are the kernels corresponding to the transform?
  - $\boldsymbol{\phi}\left(\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\end{bmatrix}\right) = \begin{bmatrix}(\boldsymbol{x_1})^3 \\ (\boldsymbol{x}_2)^3 \\ \sqrt{3}(\boldsymbol{x_1})^2\boldsymbol{x}_2 \\ \sqrt{3}(\boldsymbol{x}_2)^2\boldsymbol{x}_1\end{bmatrix}$
  - $\boldsymbol{\phi}\left(\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\end{bmatrix}\right) = \begin{bmatrix}\boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_1}\boldsymbol{x_2}\end{bmatrix}$

Replacement with a feature transformation

- For the XOR problem we defined the transformation
  - $\boldsymbol{\phi}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_1\boldsymbol{x}_2 \end{bmatrix}$
- We can thus define an inner product
  - $\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle = \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^{\boldsymbol{T}} \boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right) = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_2^{(i)}x_2^{(i)} \end{bmatrix} \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \\ x_2^{(i)}x_2^{(j)} \end{bmatrix}$

    $= x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)} + x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)}$
- This inner product implements the transformation and can potentially lead to a non-linear boundary

Kernel Functions ↔ Feature Transformation

- Since $\langle \boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{x}^{(\boldsymbol{j})} \rangle$ is always a scalar, we can actually use a function (called a kernel function $k(i, j)$) to map the two examples to a scalar value
  - Thus, the inner product from a feature transformation can be written as a kernel and a valid kernel function can thus be considered as an inner product in some feature space (proven by Moore-Aronszajn Theorem)
    - We'll talk what makes kernel valid later
  - A kernel is thus a generalized dot product
    - A measure of how similar the two examples are
      - not of whether they belong to the same class

Kernels as Similarity Functions

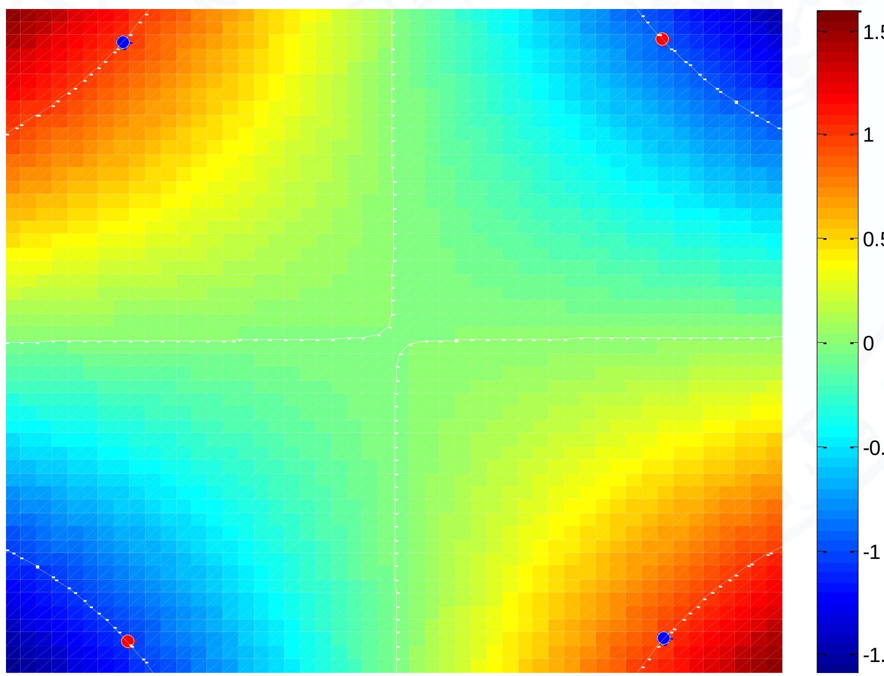
- We have been saying that kernels are similarity functions – here is how
  - Example
    - **For $\boldsymbol{K(x,y)} = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x^T y}$**
    - For the dot product, the associated distance measure is
      - $\boldsymbol{d(x,y)^2} = \|\boldsymbol{x}-\boldsymbol{y}\|^2 = (\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) = \boldsymbol{x^T x} - \boldsymbol{2x^T y} + \boldsymbol{y^T y} = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{2x^T y}$
      - This implies: $\boldsymbol{K(x,y)} = \frac{1}{2}\left(\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{d(x,y)}\right)$
      - Thus, the linear kernel measures the similarity between two points as the inverse of the square of the Euclidean distance between them (up to $\|\boldsymbol{x}\|^{\boldsymbol{2}} + \|\boldsymbol{y}\|^{\boldsymbol{2}}$)
        - » Thus the SVM with a linear kernel is no different, in its distance measurements, from a nearest neighbor classifier!!

Some kernel functions

- We can use arbitrary kernels, for example …
  - The dot-product kernel
    - $\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{b}) \ = \boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{b}$
      - Feature transform: $\boldsymbol{\phi}(\boldsymbol{a}) = \boldsymbol{a}$
  - The homogenous polynomial kernels
    - $\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{b}) = \left(\boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{b}\right)^{\boldsymbol{p}}, \boldsymbol{p}$ is called degree
      - For $\boldsymbol{p} = \mathbf{2}$ and 2D data, $\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$: $\boldsymbol{\phi}(\boldsymbol{a}) = \begin{bmatrix} a_1^2 \\ a_2^2 \\ \sqrt{2}a_1 a_2 \end{bmatrix}$
  - The Radial Basis Function (RBF) Kernel
    - $\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{e}^{-\frac{\|\boldsymbol{a}-\boldsymbol{b}\|^2}{2\boldsymbol{\sigma}^2}}$, $\boldsymbol{\sigma}$ controls the spread of the Gaussian
      - Feature transform?
- The RBF and Homogenous Polynomial kernels implement non-linear boundaries

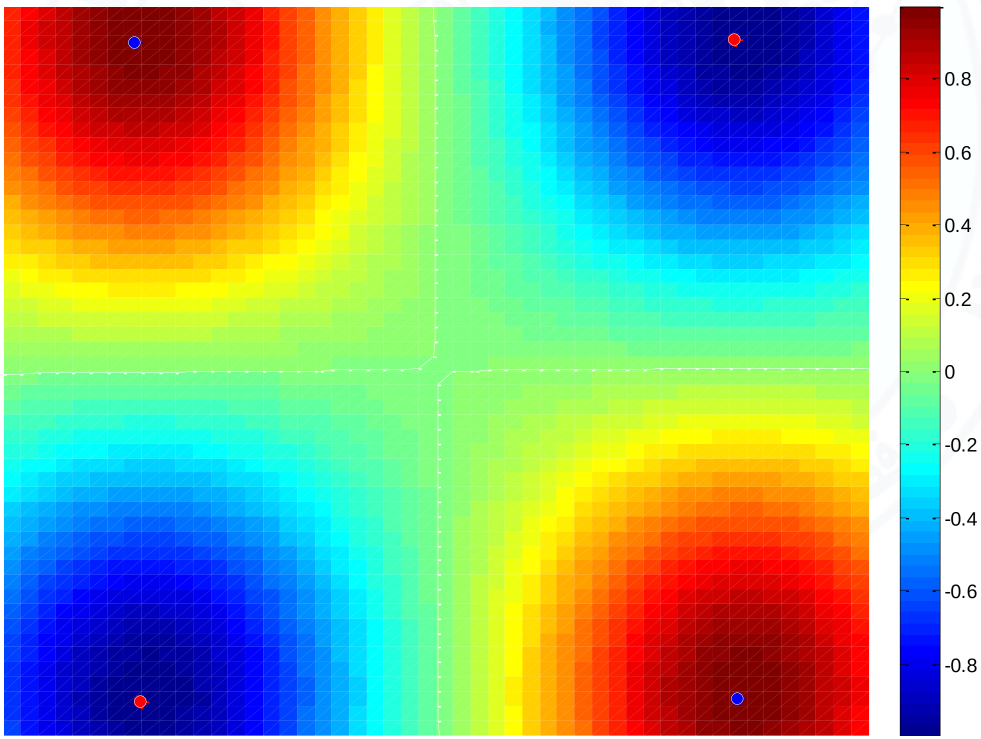
Solving the XOR

- C=1
- With polynomial kernel of degree 2



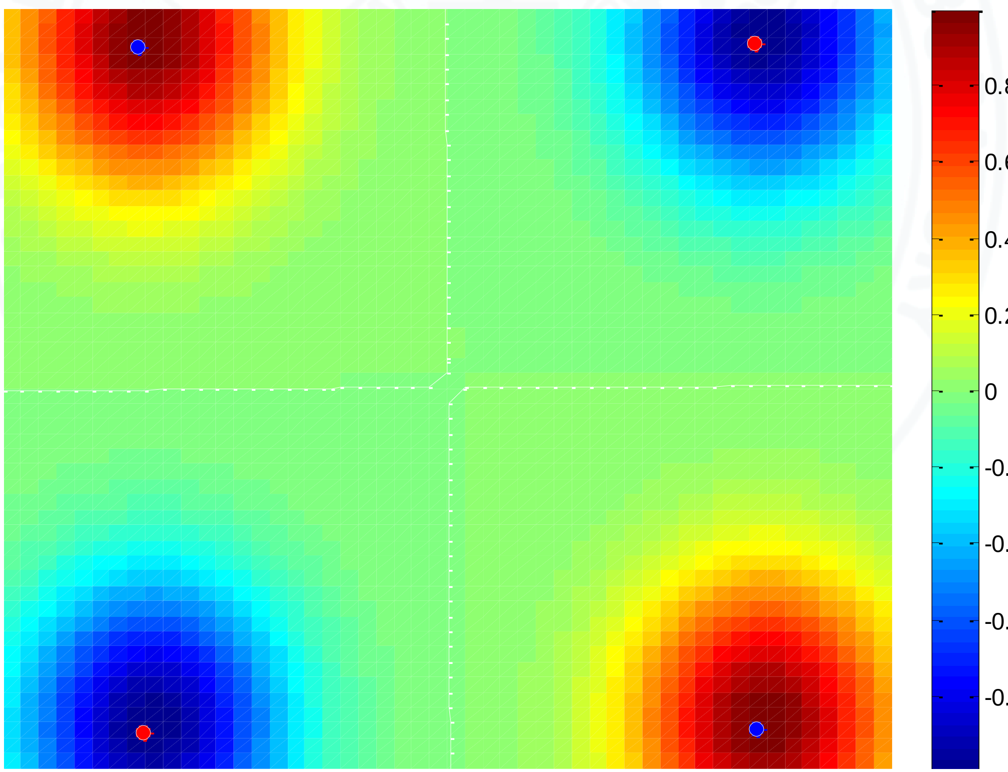
Solving the XOR

- C=1
- With RBF Kernel (sigma = 0.5)

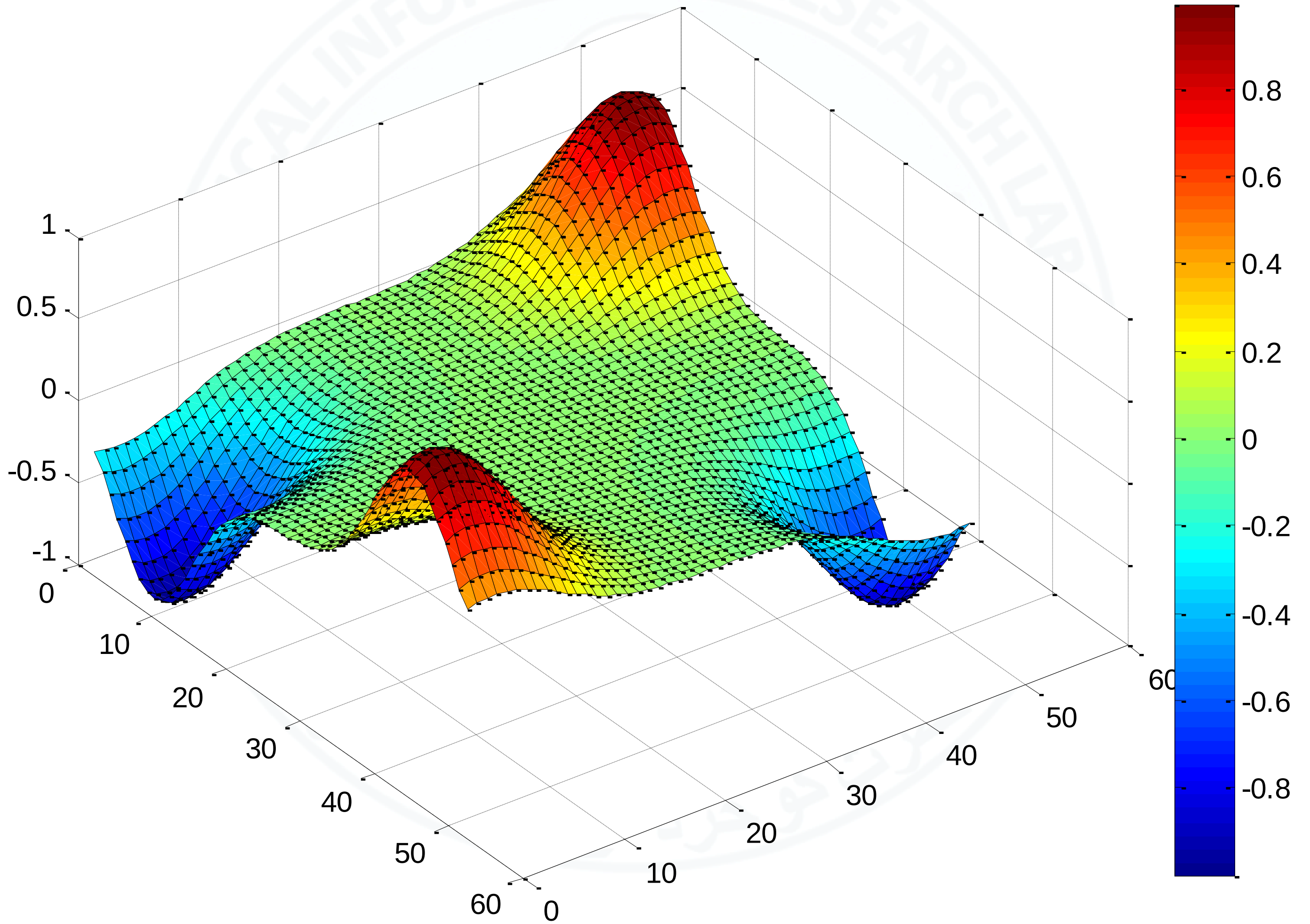


Solving the XOR

- C=1
- With RBF Kernel (sigma = 0.3)



3D Plot



1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
0
10
20
30
40
50
60
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8

Another advantage

$$\max_{\alpha_i,\beta_i\geq 0}\left\{\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\, k(i,j)\right\}$$

$$s.t \qquad 0\leq\alpha_i\leq C, \qquad \sum_{i=1}^{N}\alpha_i y_i=0$$

- Once we replace the dot product with a kernel function (i.e., perform the kernel trick or 'kernelize' the formulation), the above formulation no longer requires any features!
- As long as you have a kernel function, everything works
  - Remember a kernel function is simply a mapping from two examples to a scalar

But how is that an advantage?

- When the number of dimensions is very large, an implicit representation through a kernel is helpful
- Let's say we have a document classification problem
  - We define a M-dimensional feature vector for each document that indicates if it has any of the pre-specified number of words in it (1) or not (0)
  - M can be very large
  - The dot product of two feature vectors is equal to the number of common words between the two documents
  - Why not simply count the number of words?
    - We can now do that with the kernel trick

Kernel Trick

- You can develop a 'kernelized' version of the soft-SVM as well
- Advantage
  - Removes the explicit representation of data
  - Allows non-linear boundaries

- For understanding a 'valid' kernel, we need to introduce the concept of a kernel matrix

Kernel Matrix

- For $\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \boldsymbol{x}^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \boldsymbol{x}^{(4)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

  With the kernel: $k(i,j) = x_1^{(i)} x_1^{(j)} + x_2^{(i)} x_2^{(j)} + x_1^{(i)} x_2^{(i)} x_1^{(j)} x_2^{(j)}$
  - $k(1,1) = 0$
  - $k(1,2) = 0$
  - …
- We get this matrix
  - $K_{ij} = k(i,j)$

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	3

– If you know the kernel matrix you don’t need to know the original features anymore

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

$$\text{max.} \ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Modification Due to Kernel Function

- For testing, the new data z is classified as class 1 if *f* >0, and as class 2 if *f* <0

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

The discriminant function

- For a test object z, the discriminant function essentially is a weighted sum of the similarity between z and a pre-selected set of objects (the support vectors)

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

$\mathcal{S}$: the set of support vectors

Vectorization

- If we define the following:
  - $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_N]^{\boldsymbol{T}}$
  - $\boldsymbol{y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^{\boldsymbol{T}}$
  - $\boldsymbol{\alpha} \circ \boldsymbol{y} = [\alpha_1 y_1 \quad \alpha_2 y_2 \quad \cdots \quad \alpha_N y_N]^{\boldsymbol{T}}$
    - $\circ$ to mean element wise product
      - Hadamard product
  - $\mathbf{1}_N = [1 \quad 1 \quad \ldots \quad 1]^{\boldsymbol{T}}$
  - $\boldsymbol{X}_{(d \times N)} = [\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x}_N]$
    - **This implies:** $\boldsymbol{K} = \boldsymbol{X}^T \boldsymbol{X}$

$$max_{\boldsymbol{\alpha}}\ \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha} \circ \boldsymbol{y})^T \boldsymbol{X}^T \boldsymbol{X} (\boldsymbol{\alpha} \circ \boldsymbol{y})$$

Subject to:

$$C \succcurlyeq \boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$
$$\boldsymbol{y}^T \boldsymbol{\alpha} = 0$$

$$max_{\boldsymbol{\alpha}}\ \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha} \circ \boldsymbol{y})^T \boldsymbol{K} (\boldsymbol{\alpha} \circ \boldsymbol{y})$$

Subject to:

$$C \succcurlyeq \boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$
$$\boldsymbol{y}^T \boldsymbol{\alpha} = 0$$

Notice that there isn't any data related term here

This replacement of the dot product with the kernel is called the **kernel trick**

What is the behavior of an inner product?

- Definition of an inner product
  - Symmetry: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$
  - Linearity: $\langle \alpha \boldsymbol{x} + \beta \boldsymbol{y}, \boldsymbol{z} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \beta \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
    - Principle of superposition
  - Positive Definiteness
    - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq \mathbf{0}$
    - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \mathbf{0}$ iff $\boldsymbol{x} = \mathbf{0}$
- These conditions need to be satisfied by the kernel function too

Positive Semi Definite Kernels: Mercer's conditions

- In general a kernel is any 'similarity measure', however not every similarity function allows us to use the 'kernel trick'

- If a function k results in a symmetric positive semi definite matrix of size n x n over the given data set, then we can use the kernel trick

– $\boldsymbol{k}: \chi \times \chi \rightarrow \mathbb{R}$ is SPSD iff
  - Symmetric: $\boldsymbol{k}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{k}(\boldsymbol{x_j}, \boldsymbol{x_i})$
  - And for any choice of n (n > 0) objects $x_1, x_2, \ldots, x_n$ and any choice of real numbers $c_1, c_2, \ldots, c_N$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

Kernel Matrix

- What's the kernel matrix for this problem with the linear kernel?

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	2

- With the polynomial kernel (p=2)?

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	4

Conditions for a function to be a kernel

- A kernel function must satisfy Mercer's condition
  - The kernel matrix must be symmetric positive semi-definite
  - What does that mean?
    - $\boldsymbol{K}\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) = \boldsymbol{K}\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right)$
    - $\boldsymbol{\gamma}^T \boldsymbol{K} \boldsymbol{\gamma} \geq 0$ for any $\boldsymbol{\gamma}$
      - The Eigen-values of $\boldsymbol{K}$ must be non-negative

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	2

Dimensions: 2
Since we have 4 points, there exist labelings for which the classification problem is not linearly separable

Eigen Values
0,0,1,3

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	3

Dimensions: 3
Since we have 4 points, these points will always be separable in the corresponding feature space no matter how you label them or where you put them!

Eigen Values
0, 0.26, 1, 3.73

Example: Kernel

- $\boldsymbol{K(u,v) = e^{-\frac{\|u-v\|^2}{2\sigma^2}}}$
- The kernel matrix is (for $\boldsymbol{\sigma^2} = 0.5$):

1	e^{-1}	e^{-1}	e^{-2}
e^{-1}	1	e^{-2}	e^{-1}
e^{-1}	e^{-2}	1	e^{-1}
e^{-2}	e^{-1}	e^{-1}	1

- Eigenvalues are 1.93, 0.73, 0.47, 0.86

I get why we need symmetry but PSD?

- In the hard-SVM optimization problem we have:

$$max_{\boldsymbol{\alpha}}\ \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha} \circ \boldsymbol{y})^T \boldsymbol{K}(\boldsymbol{\alpha} \circ \boldsymbol{y})$$

Subject to:

$$\boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$
$$\boldsymbol{y}^T\boldsymbol{\alpha} = 0$$

- If we aren't careful in choosing K, this problem may have no solution

Kernel Construction

Proposition 3.22 (Closure properties) *Let* κ_1 *and* κ_2 *be kernels over* $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(\cdot)$ *a real-valued function on* X, ϕ: $X \longrightarrow \mathbb{R}^N$ *with* κ_3 *a kernel over* $\mathbb{R}^N \times \mathbb{R}^N$, *and* $\mathbf{B}$ *a symmetric positive semi-definite* $n \times n$ *matrix. Then the following functions are kernels:*

(i) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$,

(ii) $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$,

(iii) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$,

(iv) $\kappa(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$,

(v) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_3(\phi(\mathbf{x}),\phi(\mathbf{z}))$,

(vi) $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}'\mathbf{B}\mathbf{z}$.

Proposition 3.24 *Let* $\kappa_1(\mathbf{x}, \mathbf{z})$ *be a kernel over* $X \times X$, *where* $\mathbf{x}$, $\mathbf{z} \in X$, *and* $p(x)$ *is a polynomial with positive coefficients. Then the following functions are also kernels:*

(i) $\kappa(\mathbf{x}, \mathbf{z}) = p(\kappa_1(\mathbf{x}, \mathbf{z}))$,

(ii) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(\kappa_1(\mathbf{x}, \mathbf{z}))$,

(iii) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / (2\sigma^2))$.

Using the SVM

- Read:

- Ben-Hur, Asa, and Jason Weston. 2010. "A User's Guide to Support Vector Machines." In *Data Mining Techniques for the Life Sciences*, edited by Oliviero Carugo and Frank Eisenhaber, 223–39. Methods in Molecular Biology 609. Humana Press. http://dx.doi.org/10.1007/978-1-60327-241-4_13

- http://pyml.sourceforge.net/doc/howto.pdf

-

Steps for Feature based Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try

Choosing C

- Cross-validation
  - To assess how good a classifier is we can use cross-validation
    - Divide the data randomly into k parts
      - If the data is imbalanced, use stratified sampling
    - Use k-1 parts for training
    - And the held-out part for testing to evaluate accuracy or ROC curve or other performance metrics
- To choose C, you can do nested cross-validation

Handling data imbalance

- If the data is imbalanced (too much of one class and only a small number of examples from the other)
  - You can set an individual C for each example
  - Can also be used to reflect a priori knowledge

Strengths and Weaknesses of SVM

- Strengths
  - Margin maximization and kernelized
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly (through C)
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need to choose a “good” kernel function.

Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification and such SVMs do exist
- But you can also try to do multi-class classification

End of Lecture

We want to make a machine that will be proud of us.

- Danny Hillis