Kernel Methods

Simple Idea of Data Fitting

- Given $(\mathbf{x}_i, y_i)$
  - $i=1,\ldots,n$
  - $\mathbf{x}_i$ is of dimension d
- Find the best linear function $\mathbf{w}$ (hyperplane) that fits the data
- Two scenarios
  - y: real, regression
  - y: {-1,1}, classification
- Two cases
  - $n>d$, regression, least square
  - $n<d$, ridge regression
- New sample: $\mathbf{x}$, $\langle \mathbf{x},\mathbf{w} \rangle$: best fit (regression), best decision (classification)

Primary and Dual

- There are two ways to formulate the problem:
  - Primary
  - Dual
- Both provide deep insight into the problem
- Primary is more traditional
- Dual leads to newer techniques in SVM and kernel methods

Regression

$$\mathbf{w} = \arg\min_{\mathbf{w}} \left\{ \sum_i (y_i - w_o - \sum_j x_{ij} w_j)^2 \right\}$$

$$\mathbf{w} = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

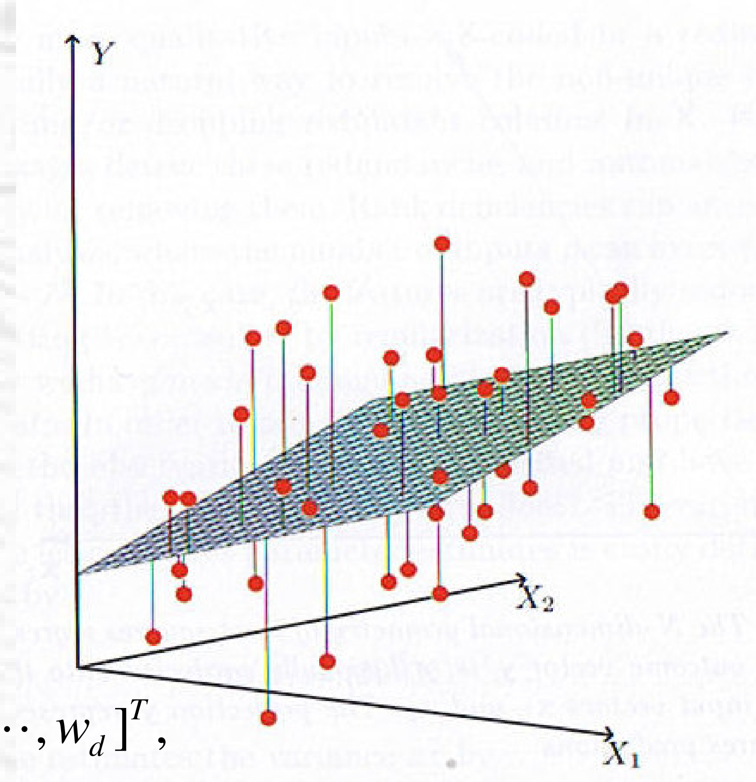
$$\frac{d(\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})}{d\mathbf{w}} = 0$$

$$\Rightarrow \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$$\Rightarrow \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$$\Rightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

$$\hat{y} = < \mathbf{x}, (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} >$$



$$\mathbf{w} = [w_o, w_1, \cdots, w_d]^T,$$

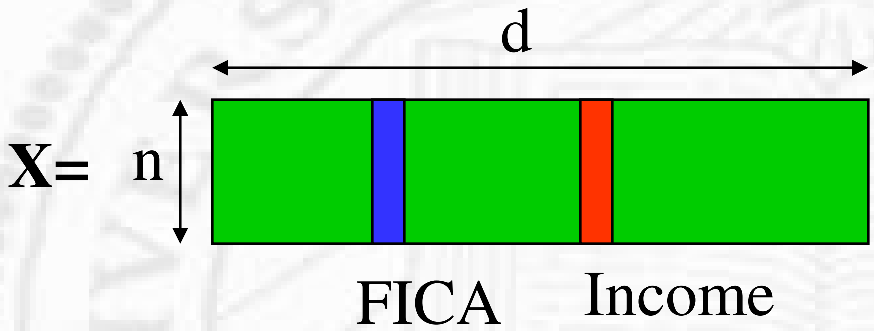
$$\mathbf{x} = [1, x_1, \cdots, x_d]^T,$$

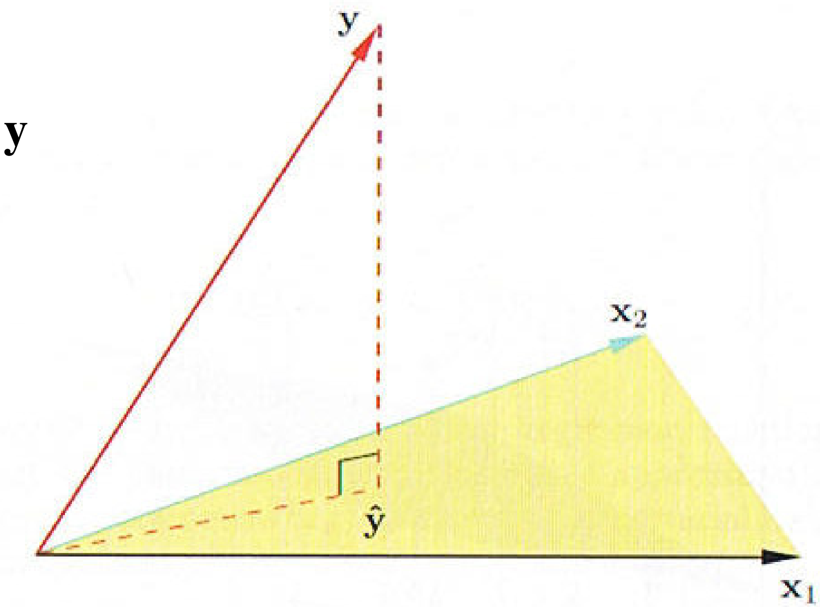
$$\mathbf{y} = [y_1, y_2, \cdots, y_n]^T$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}_{n \times xd}$$

Graphical Interpretation

$$\hat{\mathbf{y}} = \mathbf{X}\boxed{\mathbf{w}} = \boxed{\mathbf{H}}\mathbf{y} = \mathbf{X}\boxed{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}} = \boxed{\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}\mathbf{y}$$

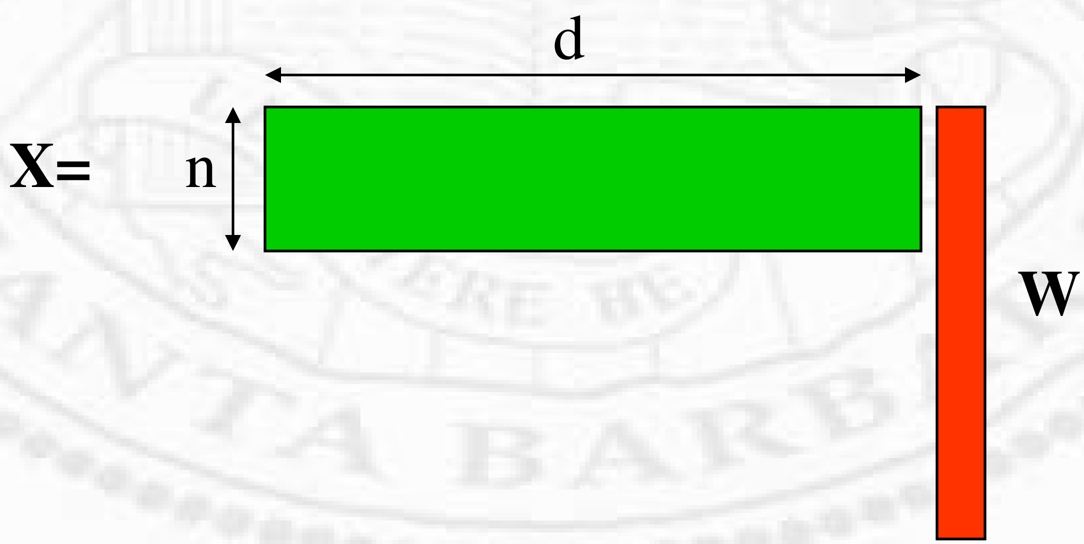




- **X** is a n (sample size) by d (dimension of data) matrix
- **w** combines the columns of **X** to best approximate **y**
  - Combine features (FICA, income, etc.) to decisions (loan)
- **H** projects **y** onto the space spanned by columns of **X**
  - Simplify the decisions to fit the features

Problem #1

- n=d, exact solution
- n>d, least square, (most likely scenarios)
- When n < d, there are not enough constraints to determine coefficients **w** uniquely



Problem #2

- If different attributes are highly correlated (income and FICA)
- The columns become dependent
- Coefficients are then poorly determined with high variance
  - E.g., large positive coefficient on one can be canceled by a similarly large negative coefficient on its correlated cousin
  - Size constraint is helpful
  - Caveat: constraint is problem dependent

Ridge Regression

- Similar to regularization

$$\mathbf{w}^{ridge} = \arg\min_{\mathbf{w}} \left\{ \sum_i (y_i - w_o - \sum_j x_{ij} w_j)^2 + \lambda \sum_j w_j^2 \right\}$$

$$\mathbf{w}^{ridge} = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

$$\frac{d(\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}}{d\mathbf{w}} = 0$$

$$\Rightarrow -\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w} = 0$$

$$\Rightarrow \mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w}$$

$$\Rightarrow \mathbf{X}^T \mathbf{y} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w}$$

$$\Rightarrow \mathbf{w}^{ridge} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

$$\Rightarrow \hat{y} = < \mathbf{x}, (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} >$$

Ugly Math

$$\mathbf{w}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \qquad \mathbf{X} = \mathbf{U}\mathbf{\Sigma}V^T$$

$$\widehat{\mathbf{y}} = \mathbf{X}\mathbf{w}^{ridge} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \mathbf{U}\mathbf{\Sigma}V^T(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}V^T + \lambda\mathbf{I})^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

$$= \mathbf{U}\mathbf{\Sigma}(V^{-T})^{-1}(\mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}V^T + \lambda\mathbf{I})^{-1}(\mathbf{V}^{-1})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

$$= \mathbf{U}\mathbf{\Sigma}(\mathbf{V}^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}V^TV^{-T} + \mathbf{V}^{-1}\lambda\mathbf{I}V^{-T})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

$$= \mathbf{U}\mathbf{\Sigma}(\mathbf{\Sigma}^T\mathbf{\Sigma} + \lambda\mathbf{I})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

$$= \sum_i \mathbf{u}_i \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\mathbf{u}_i^T\mathbf{y}$$

How to Decipher This

$$\hat{\mathbf{y}} = \sum_i \boxed{\mathbf{u}_i} \boxed{\frac{\sigma_i^2}{\sigma_i^2 + \lambda}} \boxed{\mathbf{u}_i^T \mathbf{y}}$$

- Red: best estimate (y hat) is composed of columns of **U** ("basis" features, recall **U** and **X** have the same column space)
- Green: how these basis columns are weighed
- Blue: projection of target (y) onto these columns
- Together: representing **y** in a body-fitted coordinate system ($\mathbf{u}_i$)

Sidebar

- Recall that
  - Trace (sum of the diagonals) of a matrix is the same as the sum of the eigenvalues
  - Proof: every matrix has a standard Jordan form (an upper triangular matrix) where the eigenvalues appear on the diagonal (trace=sum of eigenvalues)
  - Jordan form results from a similarity transform ($\mathbf{PAP}^{-1}$) which does not change eigenvalues

$$\mathbf{Ax} = \lambda\mathbf{x}$$

$$\Rightarrow \mathbf{PAx} = \lambda\mathbf{Px}$$

$$\Rightarrow \mathbf{PAP}^{-1}\mathbf{Px} = \lambda\mathbf{Px}$$

$$\Rightarrow \mathbf{A}^{J}\mathbf{y} = \lambda\mathbf{y}$$

Physical Interpretation

- Singular values of **X** represents the spread of data along different *body-fitting* dimensions (orthonormal columns)
- To estimate y(**=<x,w**^{ridge}**>**) regularization minimizes the contribution from less spread-out dimensions
  - Less spread-out dimensions usually have much larger variance (high dimension eigen modes) harder to estimate gradients reliably
  - Trace $\mathbf{X}(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T$ is called effective degrees of freedom

More Details

- Trace $\mathbf{X}(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T$ is called effective degrees of freedom
  - Controls how many eigen modes are actually used or active

$$df(\lambda) = d, \lambda = 0, df(\lambda) = 0, \lambda \to \infty$$

- Different methods are possible
  - Shrinking smoother: contributions are scaled
  - Projection smoother: contributions are used (1) or not used (0)

Dual Formulation

- Weight vector can be expressed as a sum of the *n* training feature vectors

$$
\begin{aligned}
\mathbf{w} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\
&= \mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-2}\mathbf{X}^T\mathbf{y} \\
&= \mathbf{X}^T{}_{d\times n}\boldsymbol{\alpha}_{n\times 1} \\
&= \sum_i \alpha_i \mathbf{x}_i
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} \\
\lambda\mathbf{w} &= \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w} \\
\mathbf{w} &= \frac{1}{\lambda}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\
&= \mathbf{X}^T{}_{d\times n}\boldsymbol{\alpha}_{n\times 1} \\
&= \sum_i \alpha_i \mathbf{x}_i
\end{aligned}
$$

Dual Formulation (cont.)

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}$$

$$\lambda\mathbf{w} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w}$$

$$\mathbf{w} = \frac{1}{\underline{\lambda}}\mathbf{X}^T\underline{(\mathbf{y} - \mathbf{X}\mathbf{w})}$$

$$= \mathbf{X}^T{}_{d\times n}\boxed{\boldsymbol{\alpha}}_{n\times 1}$$

$$= \sum_i \alpha_i \mathbf{x}_i$$

$$\boldsymbol{\alpha}_{n\times 1} = \frac{1}{\lambda}(\mathbf{y} - \mathbf{X}_{n\times d}\mathbf{w}_{d\times 1})$$

$$\lambda\boldsymbol{\alpha} = \mathbf{y} - \mathbf{X}\mathbf{w}$$

$$\lambda\boldsymbol{\alpha} = \mathbf{y} - \mathbf{X}\mathbf{X}^T\boldsymbol{\alpha}$$

$$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$$

$$\boldsymbol{\alpha} = (\mathbf{X}_{n\times d}\mathbf{X}^T{}_{d\times n} + \lambda\mathbf{I})^{-1}\mathbf{y}_{n\times 1} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

$$g(\mathbf{x}) =< \mathbf{w}, \mathbf{x} >= \left\langle \sum \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

$$=< \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}, \mathbf{x} >= \mathbf{y}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x} \rangle \\ \langle \mathbf{x}_2, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x} \rangle \end{bmatrix}$$

In More Details

Gram matrix

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}_{1\times n} \left(\begin{bmatrix} - & \mathbf{x}_1^T & - \\ - & \cdots & - \\ - & \mathbf{x}_n^T & - \end{bmatrix}_{n\times d} \begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \vdots & \mathbf{x}_n \\ | & | & | \end{bmatrix}_{d\times n} + \lambda \mathbf{I} \right)^{-1} \begin{bmatrix} - & \mathbf{x}_1^T & - \\ - & \cdots & - \\ - & \mathbf{x}_n^T & - \end{bmatrix}_{n\times d} \mathbf{x}$$

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}_{1\times n} \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 + \lambda & \mathbf{x}_1^T\mathbf{x}_2 & . & \mathbf{x}_1^T\mathbf{x}_n \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 + \lambda & . & \mathbf{x}_2^T\mathbf{x}_n \\ . & . & \ddots & . \\ \mathbf{x}_n^T\mathbf{x}_1 & \mathbf{x}_n^T\mathbf{x}_2 & & \mathbf{x}_n^T\mathbf{x}_n + \lambda \end{bmatrix} \begin{bmatrix} - & \mathbf{x}_1^T\mathbf{x} & - \\ - & \cdots & - \\ - & \mathbf{x}_n^T\mathbf{x} & - \end{bmatrix}_{n\times 1}$$

Observations

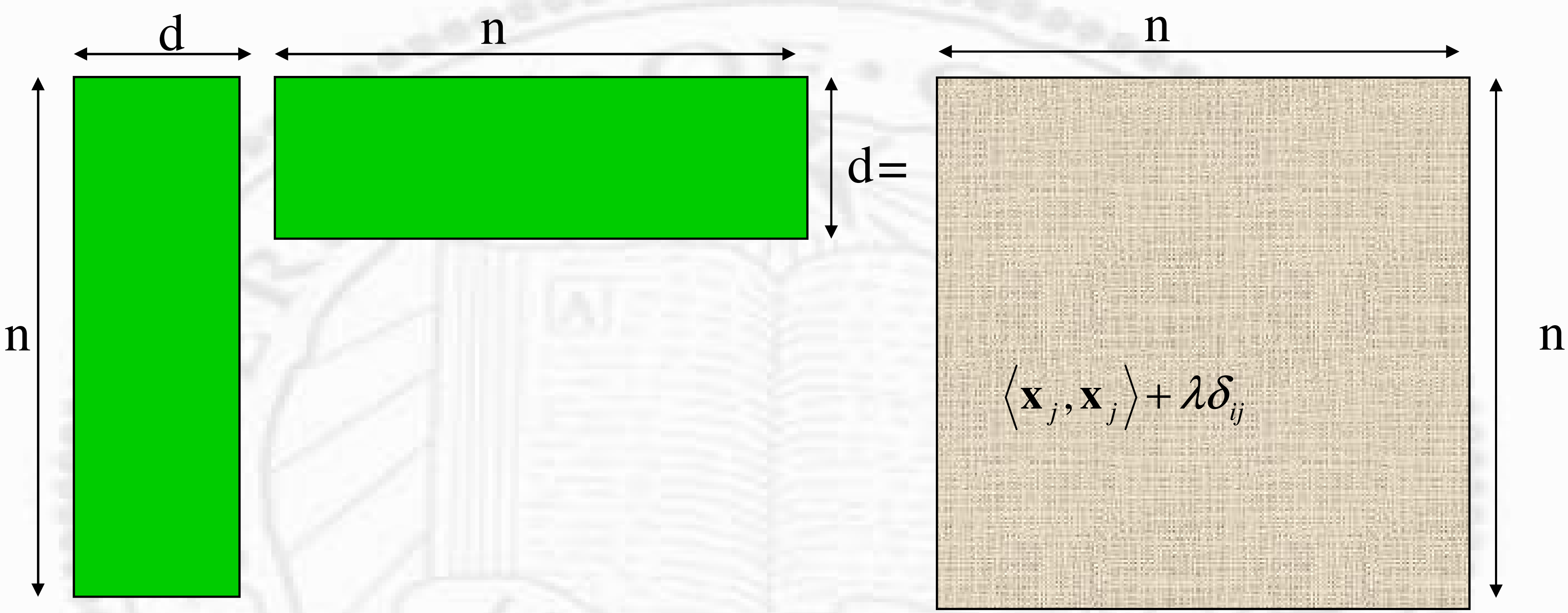
- Primary
- $\mathbf{X}^T\mathbf{X}$ is d by d
- Training: Slow for high feature dimension
- Use: fast O(d)

$$g(\mathbf{x}) = < \mathbf{x}_{d\times 1}, (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}_{d\times 1} >$$

- Dual
- Only inner products are involved
- $\mathbf{X}\mathbf{X}^T$ is n by n
- Training: Fast for high feature dimension
- Use: Slow O(nd)
  - N inner product to evaluate, each requires d multiplications

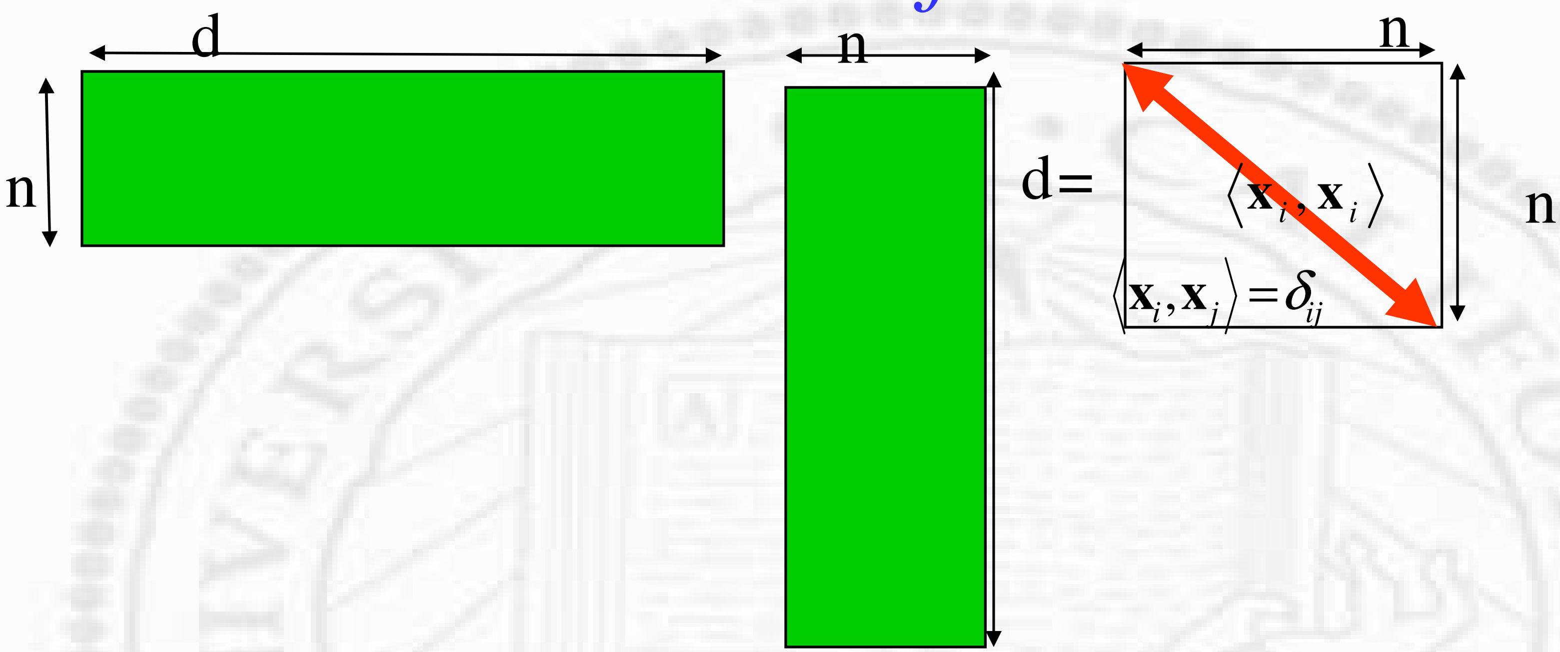
$$g(\mathbf{x}) = \mathbf{y}^T(\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1}{}_{1\times n} \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x} \rangle \\ \langle \mathbf{x}_2, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x} \rangle \end{bmatrix}_{n\times 1}$$

Graphical Interpretation



$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle = \mathbf{y}^T (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1} \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x} \rangle \\ \langle \mathbf{x}_2, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x} \rangle \end{bmatrix}$$

One Extreme – Perfect Uncorrelated



$$g(\mathbf{x}) = \mathbf{y}^T (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1} \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x} \rangle \\ \langle \mathbf{x}_2, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x} \rangle \end{bmatrix} = \sum_i y_i \frac{\langle \mathbf{x}_i, \mathbf{x} \rangle}{\langle \mathbf{x}_i, \mathbf{x}_i \rangle + \lambda}$$

- Orthogonal projection – no generalization

General Case

$$\hat{\mathbf{y}}^T{}_{1\times n} = \mathbf{y}^T{}_{1\times n}(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I})^{-1}{}_{n\times n}\mathbf{X}_{n\times d}\mathbf{X}^T{}_{d\times n} \qquad \mathbf{X}=\mathbf{U}_{n\times d}\boldsymbol{\Sigma}_{d\times d}V^T{}_{d\times d}$$

$$= \mathbf{y}^T{}_{1\times n}(\mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T+\lambda\mathbf{I})^{-1}{}_{n\times n}\mathbf{U}\boldsymbol{\Sigma}V^T\mathbf{X}^T{}_{d\times n}$$

$$= \mathbf{y}^T{}_{1\times n}(\mathbf{U}(\boldsymbol{\Sigma}^2+\lambda\mathbf{I})\mathbf{U}^T)^{-1}{}_{n\times n}\mathbf{U}\boldsymbol{\Sigma}V^T\mathbf{X}^T{}_{d\times n}$$

$$= \mathbf{y}^T{}_{1\times n}\mathbf{U}(\boldsymbol{\Sigma}^2+\lambda\mathbf{I})^{-1}\mathbf{U}^{-1}\mathbf{U}\boldsymbol{\Sigma}V^T\mathbf{X}^T{}_{d\times n}$$

$$= \mathbf{y}^T{}_{1\times n}\mathbf{U}(\boldsymbol{\Sigma}^2+\lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}V^T\mathbf{X}^T{}_{d\times n}$$

$$= (\mathbf{U}^T{}_{d\times n}\mathbf{y}_{n\times 1})^T{}_{1\times d}(\boldsymbol{\Sigma}^2+\lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}V^T\boxed{\mathbf{X}^T{}_{d\times n}}$$

$$= (\mathbf{U}^T{}_{d\times n}\mathbf{y}_{n\times 1})^T{}_{1\times d}(\boldsymbol{\Sigma}^2+\lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}V^T\boxed{V\boldsymbol{\Sigma}\mathbf{U}^T}$$

$$= (\mathbf{U}^T{}_{d\times n}\mathbf{y}_{n\times 1})^T{}_{1\times d}(\boldsymbol{\Sigma}^2+\lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}^2\mathbf{U}^T$$

- How to interpret this? Does this still make sense?

Physical Meaning of SVD

- Assume that $n > d$
- $\mathbf{X}$ is of rank d at most
- $\mathbf{U}$ are the body (data)-fitted axes
- $\mathbf{U}^T$ is a projection from n to d space
- Σ is the importance of the dimensions
- **V** is the representation of the **X** in the d space

$$\mathbf{X} = \mathbf{U}_{n \times d} \mathbf{\Sigma}_{d \times d} V^T{}_{d \times d}$$

Interpretation

$$\hat{\mathbf{y}}^T{}_{1\times n} = (\mathbf{U}^T{}_{d\times n}\mathbf{y}_{n\times 1})^T{}_{1\times d}(\boldsymbol{\Sigma}^2 + \lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}^2\mathbf{U}^T \longrightarrow \hat{\mathbf{y}} = \sum_i \mathbf{u}_i \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\mathbf{u}_i^T\mathbf{y}$$

- In the new, uncorrelated space, there are only *d* training vectors and *d* decisions
- Red: *dx1* uncorrelated decision vector
- Green: weighting of the significance of the components in the uncorrelated decision vector
- Blue: transformed (uncorrelated) training samples
- Still the same interpretation: similarity measurement in a new space by
  - Gram matrix
  - Inner product of training samples and new sample

First Important Concept

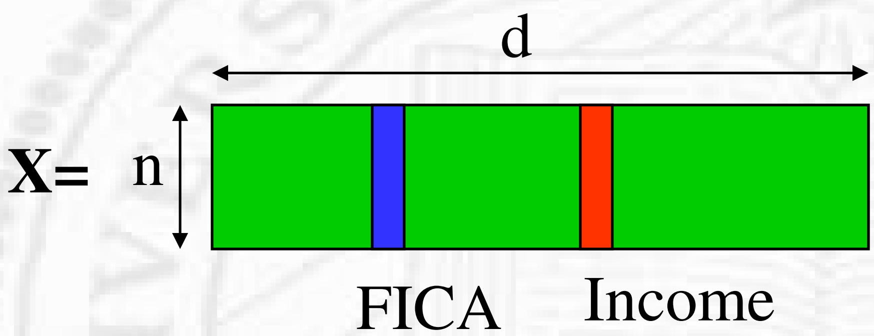
- The computation involves *only* inner product
  - For training samples in computing the Gram matrix
  - For new sample in computing regression or classification results
- Similarity is measured in terms of *angle*, instead of *distance*

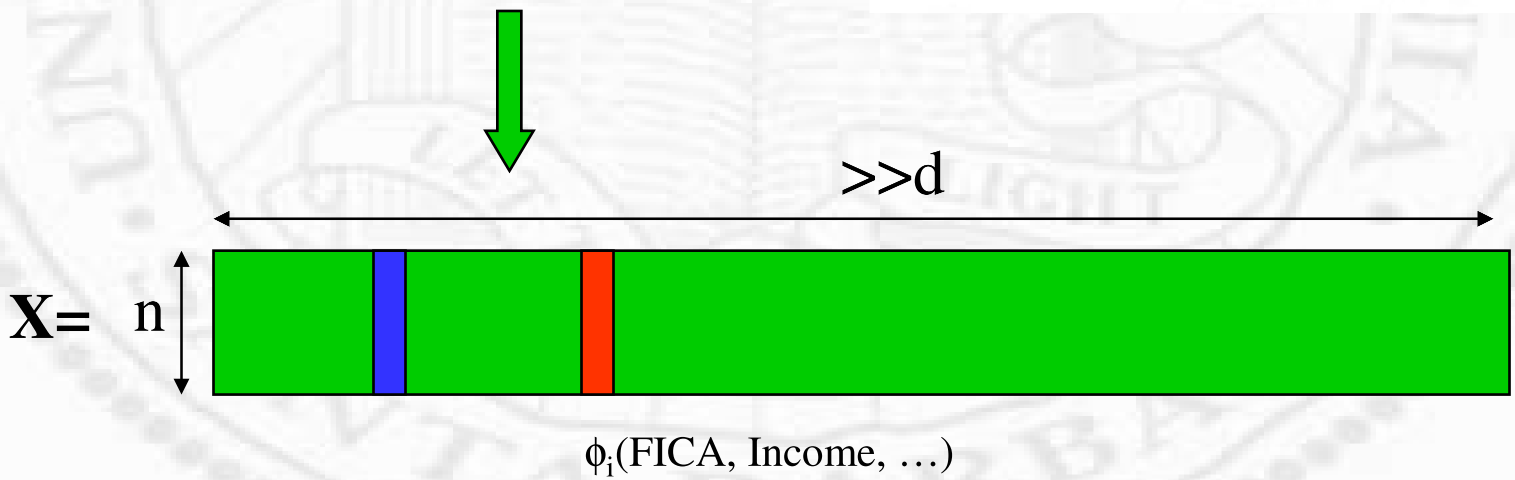
Second Important Concept

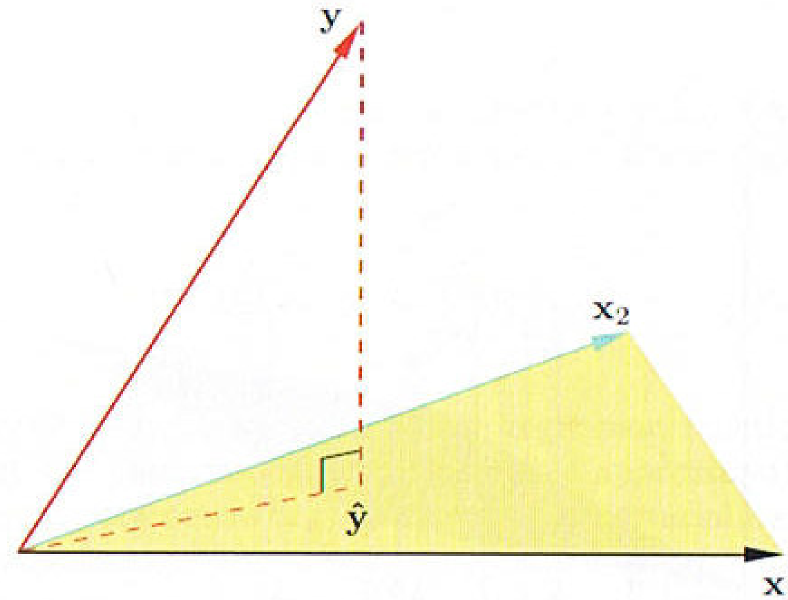
- Using angle or distance for similarity measurement doesn't make problems easier or harder
  - If you cannot separate data, it doesn't matter what similarity measures you use
- "Massage" data
  - Transform data (into higher – even infinite - dimensional space)
  - Data become "more likely" to be linearly separable (caveat: choice of the kernel function is important)
  - Cannot perform inner product efficiently
  - Kernel trick – do not have to

Why?

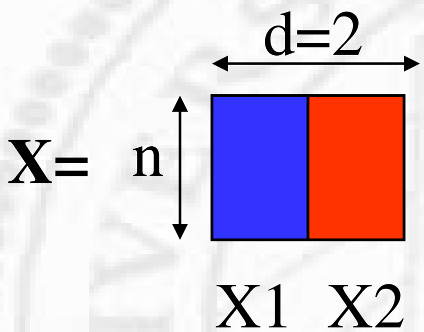
- There are a lot more “bases” features now

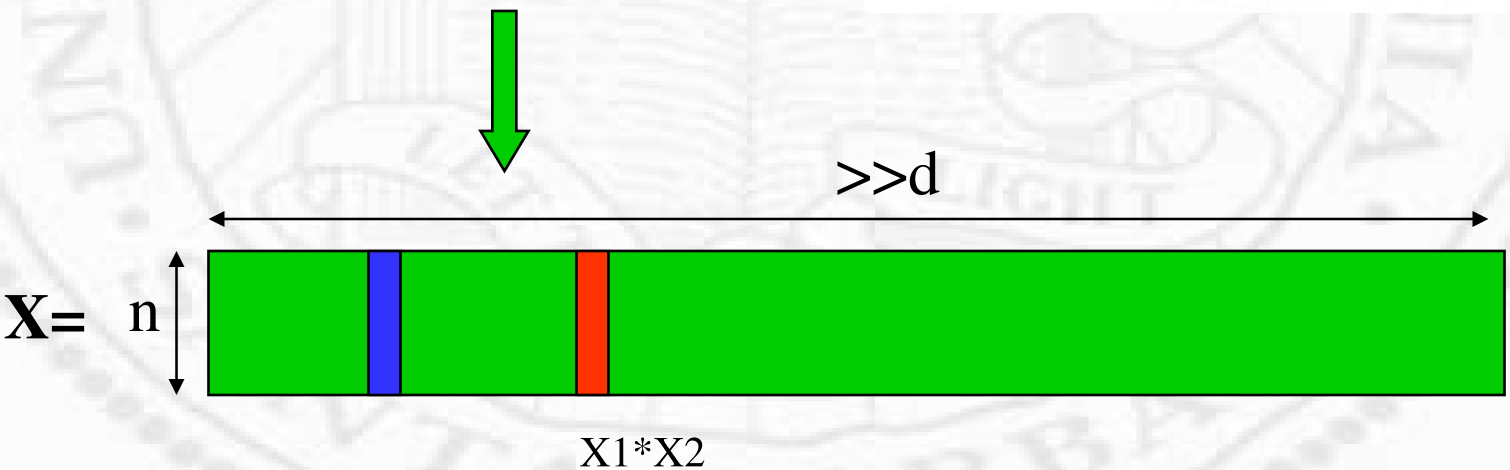


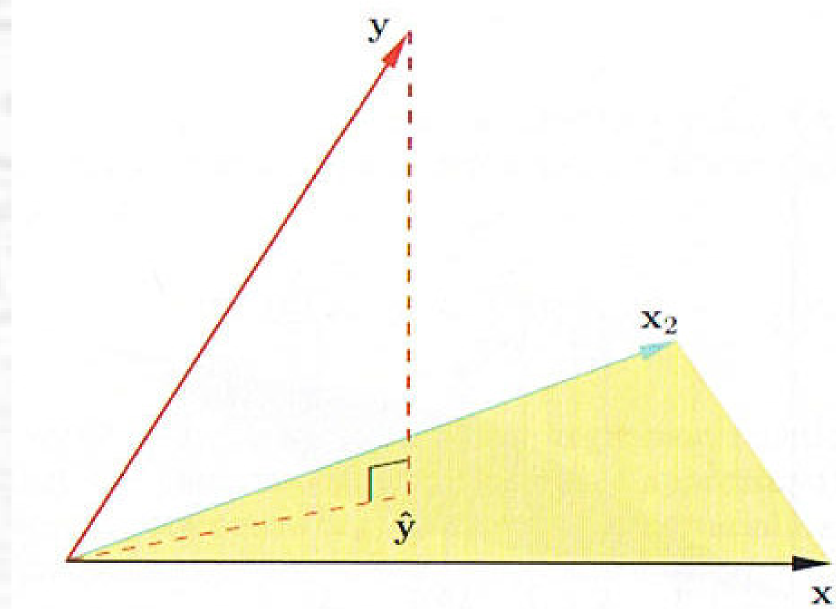




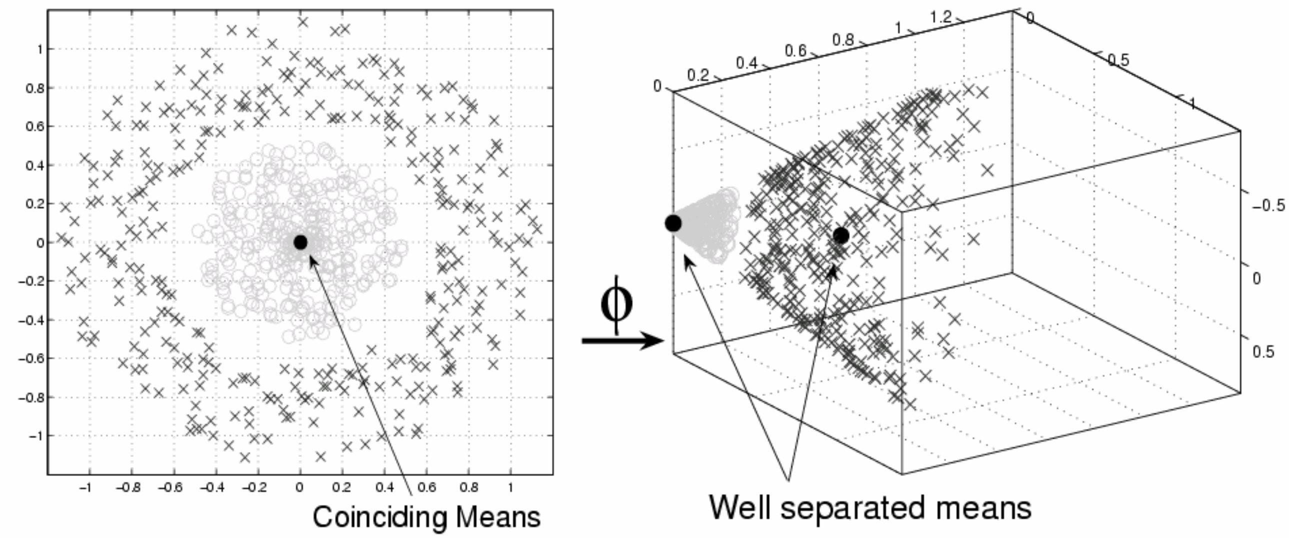
Example - Xor







Example (Doesn't quite work yet)



(a) Input Space (b) Feature Space

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1, x_2, x_1^2 + x_2^2) \in F = R^3$$

- Need to keep the nice property of requiring only inner product in the computation (dual formulation)
- But what happens if the feature dimension is very high (or even infinitely high)?
- Inner product in the high (infinitely high) dimensional feature space can be calculated without explicit mapping through a kernel function

In More Details

$$g(\mathbf{x}) = < \mathbf{w}, \mathbf{x} > = \left\langle \sum \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle = \mathbf{y}^T (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1} \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x} \rangle \\ \langle \mathbf{x}_2, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x} \rangle \end{bmatrix}$$

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}_{1\times n} \left(\begin{bmatrix} - & \phi(\mathbf{x}_1)^T & - \\ - & \cdots & - \\ - & \phi(\mathbf{x}_n)^T & - \end{bmatrix}_{n\times d} \begin{bmatrix} | & | & | \\ \phi(\mathbf{x}_1) & \vdots & \phi(\mathbf{x}_n) \\ | & | & | \end{bmatrix}_{d\times n} + \lambda \mathbf{I} \right)^{-1} \begin{bmatrix} - & \phi(\mathbf{x}_1)^T & - \\ - & \cdots & - \\ - & \phi(\mathbf{x}_n)^T & - \end{bmatrix}_{n\times d} \phi(\mathbf{x})$$

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}_{1\times n} \begin{bmatrix} \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_1) + \lambda & \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) & . & \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_n) \\ \phi(\mathbf{x}_2)^T \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2)^T \phi(\mathbf{x}_2) + \lambda & . & \phi(\mathbf{x}_2)^T \mathbf{x}_n) \\ . & . & \ddots & . \\ \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_1) & \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_2) & & \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_n) + \lambda \end{bmatrix} \begin{bmatrix} - & \phi(\mathbf{x}_1)^T \phi(\mathbf{x}) & - \\ - & \cdots & - \\ - & \phi(\mathbf{x}_n)^T \phi(\mathbf{x}) & - \end{bmatrix}_{n\times}$$

Example

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, 2x_1x_2, x_2^2) \in F = R^3$$

$$k(\mathbf{x,y}) = (x_1 y_1 + x_2 y_2)^2$$

$$= x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1 y_2, y_2^2)$$

$$= \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

More Example

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, x_1 x_2, x_1 x_2, x_2^2) \in F = R^4$$

$$k(\mathbf{x,y}) = (x_1 y_1 + x_2 y_2)^2$$

$$= x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + x_2^2 y_2^2$$

$$= (x_1^2, x_1 x_2, x_1 x_2, x_2^2) \cdot (y_1^2, y_1 y_2, y_1 y_2, y_2^2)$$

$$= \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

Even More Example

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = \frac{1}{\sqrt{2}}(x_1^2 - x_2^2, 2x_1x_2, x_1^2 + x_2^2) \in F = R^3$$

$$k(\mathbf{x,y}) = (x_1y_1 + x_2y_2)^2$$

$$= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$= \frac{1}{\sqrt{2}}(x_1^2 - x_2^2, 2x_1x_2, x_1^2 + x_2^2) \cdot \frac{1}{\sqrt{2}}(y_1^2 - y_2^2, 2y_1y_2, y_1^2 + y_2^2)$$

$$= \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

- The interpretation of mapping ϕ is not unique even with a single κ function

Observations

- The interpretation of mapping ϕ is not unique even with a single κ function
- The κ function is special. Certainly not all functions have such properties (i.e., corresponding to the inner product in a feature space)
- Such functions are called kernel functions
  - Kernel is a function that for all $\mathbf{x}$, $\mathbf{z}$ in X, $\kappa(\mathbf{x}, \mathbf{z}) = \langle\phi(\mathbf{x}), \phi(\mathbf{z})\rangle$, where ϕ is a mapping from X to an (inner product) feature space F

Important Theorem

- A function κ: $\kappa(X,X)\text{-}> R$ can be decomposed into $\kappa(x,z)\text{-}> <\phi(x),\phi(z)>$ (ϕ forms a Hilbert space) if and only if it satisfies *finitely* positive semi-definite property
  - *Finitely* positive semi-definite: If $\kappa(X,X)\text{-}> R$ is symmetrical and for ***any*** finite subset of space X, the matrix formed by applying κ is positive semi-definite (i.e., Gram matrix is SPD for any choices of training samples)

Only if Condition

- Given bi-linear function κ: κ(X,X)-> R κ(x,z)-> <ϕ(x),ϕ(z)> then the Gram matrix from κ satisfies *finitely* positive semi-definite property

$$\mathbf{v}^T\mathbf{G}\mathbf{v} = \mathbf{v}^T \begin{bmatrix} \kappa(x_1,x_1) & \kappa(x_1,x_2) & \cdots & \kappa(x_1,x_n) \\ \kappa(x_2,x_1) & \kappa(x_2,x_2) & \cdots & \kappa(x_2,x_n) \\ \cdots & \cdots & \cdots & \cdots \\ \kappa(x_n,x_1) & \kappa(x_n,x_2) & \cdots & \kappa(x_n,x_n) \end{bmatrix} \mathbf{v}$$

$$= \mathbf{v}^T \begin{bmatrix} \phi(\mathbf{x}_1) \\ \phi(\mathbf{x}_2) \\ \vdots \\ \phi(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix} \mathbf{v}$$

$$= \sum v_i \phi(\mathbf{x}_i) \sum v_i \phi(\mathbf{x}_i) = \left\| \sum v_i \phi(\mathbf{x}_i) \right\|^2 \geq 0$$

If Condition Proof Strategy

- More complicated
- First: Establish a Hilbert (*function*) space
- Second: Establish the reproducing property in this function space
- Third: Establish the (Fourier) basis of such a function space
- Fourth: Establish κ as expansion on such a Fourier basis

What?

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, 2x_1x_2, x_2^2) \in F = R^3$$

$$k(\mathbf{x}, \mathbf{y}) = (x_1y_1 + x_2y_2)^2$$

$$= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$= \boxed{\phi}(\mathbf{x}) \cdot \boxed{\phi}(\mathbf{y})$$

- In this case, ϕ is a 3 dimensional space
- Each "dimension" is a function of **x**
- There are three (not unique) "eign" functions that form the basis

A Function Space Example

- All (well-behaved, square-integrable) functions defined over a domain *R* form a vector space (a function space, $\mathcal{F}$)
  - $f \in \mathcal{F}$, then $cf \in \mathcal{F}$
  - $f \in \mathcal{F}$, and $g \in \mathcal{F}$, then $(af+bg) \in \mathcal{F}$
- Such a space is a Hilbert space if it is complete with an inner product (real valued, symmetrical, bilinear)

$$< f, g >=< g, f >= \int f(x) g(x) dx$$

$$< f, f >= \int f(x) f(x) dx = \int f^2(x) dx > 0$$

- You can define an orthogonal basis (e.g., Fourier basis) on it

Hilbert Space

- The proof is harder and not very intuitive
- Suffice it to say that someone has figured out that the desired feature space is a *function* space of the form

$$F = \left\{ \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, .) : l \in N, \mathbf{x}_i \in \mathbf{X}, \alpha_i \in R, i = 1, \cdots, l \right\}$$

- With an inner product defined as

$$F = \left\{ \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, .) : l \in N, \mathbf{x}_i \in \mathbf{X}, \alpha_i \in R, i = 1, \cdots, l \right\}$$

$$f = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, .), \quad g = \sum_{i=1}^{m} \beta_i k(\mathbf{z}_i, .)$$

$$< f, g >= \sum_{j=1}^{m} \sum_{i=1}^{l} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{z}_j) = \sum_{i=1}^{l} \alpha_i g(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j f(\mathbf{z}_j)$$

Why

- Because then we have SPD properties regardless of choice of $\mathbf{x}_i$

$$F = \left\{ \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i,.) : l \in N, \mathbf{x}_i \in \mathbf{X}, \alpha_i \in R, i = 1, \cdots, l \right\}$$

$$f = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i,.), \quad g = \sum_{i=1}^{m} \beta_i k(\mathbf{z}_i,.)$$

$$< f, g >= \sum_{j=1}^{m} \sum_{i=1}^{l} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{z}_j) = \sum_{i=1}^{l} \alpha_i g(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j f(\mathbf{z}_j)$$

$$< f, f >= \sum_{j=1}^{l} \sum_{i=1}^{l} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{z}_j) = \boldsymbol{\alpha}^T \mathbf{k} \boldsymbol{\alpha} \geq 0$$

- Still have to prove completeness (not here, see page 62 of Shawe-Taylor and Christianini)

Reproducing Property

- Special Hilbert space called Reproducing Kernel Hilbert space (RKHS)

$$\text{Recall}: \; f = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i,.), \quad g = \sum_{i=1}^{m} \beta_i k(\mathbf{z}_i,.)$$

$$< f, g >= \sum_{j=1}^{m} \sum_{i=1}^{l} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{z}_j) = \sum_{i=1}^{l} \alpha_i g(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j f(\mathbf{z}_j)$$

If we take $g = k(\mathbf{x},.)$

$$< f, g >=< f, k(\mathbf{x},.) >= \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x})$$

Mercer Kernel Theorem

- Denote an orthonormal basis of the RKHS with kernel κ as $\phi_i(.)$
- κ(x,.) belongs in this space
- Expand κ(x,.) onto the orthonormal basis $\phi_i(.)$

$$k(\mathbf{x},.) = \sum_{i=1}^{\infty} < k(\mathbf{x},.), \phi_i(.) > \phi_i(.)$$

$$\Rightarrow k(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{\infty} < k(\mathbf{x},\mathbf{z}), \phi_i(\mathbf{z}) > \phi_i(\mathbf{z})$$

$$\Rightarrow k(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{\infty} \underline{< k(\mathbf{x},\mathbf{z}), \phi_i(\mathbf{z}) >} \phi_i(\mathbf{z})$$

$$\Rightarrow k(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{\infty} \underline{\phi_i(\mathbf{x})}\phi_i(\mathbf{z})$$

Reproducing property

Practically

- The explicit computation of feature mapping is *not* necessary
- Instead, we can compose different κ and manipulate Gram matrices using all kinds of mathematical tricks (Kernel design), as long as the finite positive definite property is preserved
- Research intensive topic, not covered in detail here

Composition of Kernels

- The space of kernel functions is closed under certain operations
  - I.e., the composition of valid kernel functions using such operations result in valid kernels
  - Can be proven by showing the resulting function preserves the finite positive definite property
  - E.g., sum and multiplication of kernels, and constant multiplication by a positive number

$$\boldsymbol{\alpha}^T \mathbf{k}_1 \boldsymbol{\alpha} > 0 \quad \boldsymbol{\alpha}^T \mathbf{k}_2 \boldsymbol{\alpha} > 0$$

$$\boldsymbol{\alpha}^T c\mathbf{k}_1 \boldsymbol{\alpha} = c\boldsymbol{\alpha}^T \mathbf{k}_1 \boldsymbol{\alpha} > 0, if \quad c > 0$$

$$\boldsymbol{\alpha}^T (\mathbf{k}_1 + \mathbf{k}_2)\boldsymbol{\alpha} = \boldsymbol{\alpha}^T \mathbf{k}_1 \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{k}_2 \boldsymbol{\alpha} > 0$$

Other Rules

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$$

$$k(\mathbf{x}, \mathbf{z}) = ak_1(\mathbf{x}, \mathbf{z}),\ a > 0$$

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z})$$

$$\Rightarrow\ k(\mathbf{x}, \mathbf{z}) = p(k_1(\mathbf{x}, \mathbf{z}))$$

$$\Rightarrow\ k(\mathbf{x}, \mathbf{z}) = \exp(k_1(\mathbf{x}, \mathbf{z}))$$

$$\Rightarrow\ k(\mathbf{x}, \mathbf{z}) = \exp(k_1(\mathbf{x} - \mathbf{z})/(2\sigma^2))$$

Kernel Shaping

- Adding a constant to all entries
  - Adding an extra constant features
- Adding a constant to diagonal
  - Ridge regression, drop smaller features

Example Kernels

- Pattern classification is a hard problem
- Massaging classifiers is difficult and massaging data (using different kernels) only allocate the complexity differently (you cannot turn a NP problem into a P problem by a magic trick)
- Whether Kernel methods work will depend on your kernels
- Some examples are discussed below (there are many more …)

Polynomial Kernel

$$k(\mathbf{x},\mathbf{z}) = p(k_1(\mathbf{x},\mathbf{z})) = (<\mathbf{x},\mathbf{z}> + c)^d$$

$$\text{(if feature is two dimensional and d = 2)}$$

$$= (x_1 z_1 + x_2 z_2 + c)^2$$

$$= (x_1 z_1 + x_2 z_2)^2 + 2c(x_1 z_1 + x_2 z_2) + c^2$$

$$= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 + 2c x_1 z_1 + 2 x_2 z_2 + c^2$$

$$= (x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2c}\, x_1, \sqrt{2c}\, x_2, c) \cdot (z_1^2, \sqrt{2} z_1 z_2, z_2^2, \sqrt{2c}\, z_1, \sqrt{2c}\, z_2, c)$$

$$= \phi(\mathbf{x}).\phi(\mathbf{z}) = \sum_i \phi_i(\mathbf{x}).\phi_i(\mathbf{z})$$

- The feature space is made of all monomials of the form

$$x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}, \sum_{j=1}^{n} i_j = s, s \le d$$

- The dimension is $\begin{pmatrix} n+d \\ d \end{pmatrix}$

- Instead of calculating so many terms, we can do a simple polynomial evaluation – the beauty of kernel methods (less control over weighting of individual monomials)

All-Subsets Kernel

If there are n features, 1, ..., n

$\phi_A, A \subseteq \{1,2,\ldots,n\}$

$$\phi_A(\mathbf{x}) = \prod_{j \in A} x_j^{i_j} = x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \quad \sum_{j=1}^{n} i_j \le n, i_j \in \{0,1\}, 1 \le j \le n$$

- The feature space is made of all monomials of the form

$$\prod_{i \in A} x_j^{i_j} = x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \quad \sum_{j=1}^{n} i_j \le n, i_j \in \{0,1\}, 1 \le j \le n$$

- The dimension is 2^n

All-Subsets Kernel (cont.)

- Instead of calculating so many terms, we can do a simple polynomial evaluation

If there are n features, 1, ..., n

$\phi_A, A \subseteq \{1,2,...,n\}$

$$\kappa(\mathbf{x},\mathbf{z}) = <\phi(\mathbf{x}),\phi(\mathbf{z})> = \sum_{A\subseteq\{1,2,...,n\}} \phi_A(\mathbf{x})\phi_A(\mathbf{z}) = \prod_{i=1}^{n}(1+x_i z_i)$$

More generally

$$\kappa(\mathbf{x},\mathbf{z}) = <\phi(\mathbf{x}),\phi(\mathbf{z})> = \sum_{A\subseteq\{1,2,...,n\}} \phi_A(\mathbf{x})\phi_A(\mathbf{z}) = \prod_{i=1}^{n}(1+\boxed{a_i} x_i z_i)$$

Different weights for different features

ANOVA Kernel

- All-subset kernel of a fixed cardinality *d*
- Dimensionality is $\binom{n}{d}$

If there are n features, 1, ..., n

$$\phi_A, A \subseteq \{1,2,\ldots,n\}, \boxed{|A| = d}$$

$$\phi_A(\mathbf{x}) = \prod_{j \in A} x_j^{i_j} = x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \quad \boxed{\sum_{j=1}^{n} i_j = d}, i_j \in \{0,1\}, 1 \le j \le n$$

- Evaluation through recursion (DP)

Gaussian Kernel

- Identical to the Radial Basis Function

$$\kappa(\mathbf{x},\mathbf{z}) = <\phi(\mathbf{x}),\phi(\mathbf{z})> = \exp(\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2})$$

- Recall that $$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

- The feature dimension is infinitely high in this case

Representing Texts

- Bag-of-words model
  - Presence + frequency
  - Ordering, grammatical relations, phrases ignored
  - Terms: words
  - Dictionary: all possible words
  - Corpses: all documents
  - Document

$$d \rightarrow \phi(d) = (tf(t_1, d), tf(t_2, d), \cdots, f(t_k, d)) \in R^k$$

  - Similarity is measured by the inner product of $\phi(d)$

Mapping between terms and docs

- Document-term matrices (**D**)
  - **X** in our previous notation
- Term-document matrices (**D**^{T})
  - **X**' in our previous notation
- Document-document matrices (**D D**^{T})
  - Gram matrix, dual formulation
- Term-term matrices (**D**^{T} **D**)
  - Primary formulation

$$d \rightarrow \phi(d) = (tf(t_1, d), tf(t_2, d), \cdots, f(t_k, d)) \in R^k$$

$$\mathbf{D}(\mathbf{X}) = \begin{bmatrix} tf(t_1, d_1) & tf(t_2, d_1) & . & tf(t_k, d_1) \\ tf(t_1, d_2) & tf(t_2, d_2) & . & tf(t_k, d_2) \\ & & \ddots & \\ tf(t_1, d_n) & tf(t_2, d_n) & . & tf(t_k, d_n) \end{bmatrix}$$

Strings and Sequences

- DNA, protein, virus signatures, etc.
  - Different lengths
  - Partial matching
  - Multiple matched sub-regions
  - Good example of kernels on non-numerical data set
  - Dynamic programming (DP) is the standard (expensive) matching technique to define similarity

Spectrum Kernels

- *p*-spectrum: histogram of (contiguous) substring of length *p*
- Kernel as inner product of p-spectrum of two

Example 11.8 [2-spectrum kernel] Consider the strings "bar", "bat", "car" and "cat". Their 2-spectra are given in the following table:

ϕ	ar	at	ba	ca
bar	1	0	1	0
bat	0	1	1	0
car	1	0	0	1
cat	0	1	0	1

with all the other dimensions indexed by other strings of length 2 having value 0, so that the resulting kernel matrix is:

K	bar	bat	car	cat
bar	2	1	1	0
bat	1	2	0	1
car	1	0	2	1
cat	0	1	1	2

All Subsequences Kernel

Example 11.16 All the (non-contiguous) subsequences in the words "bar", "baa", "car" and "cat" are given in the following two tables:

ϕ	ε	a	b	c	r	t	aa	ar	at	ba	br	bt
bar	1	1	1	0	1	0	0	1	0	1	1	0
baa	1	2	1	0	0	0	1	0	0	2	0	0
car	1	1	0	1	1	0	0	1	0	0	0	0
cat	1	1	0	1	0	1	0	0	1	0	0	0

ϕ	ca	cr	ct	bar	baa	car	cat
bar	0	0	0	1	0	0	0
baa	0	0	0	0	1	0	0
car	1	1	0	0	0	1	0
cat	1	0	1	0	0	0	1

and since all other (infinite) coordinates must have value zero, the kernel matrix is

K	bar	baa	car	cat
bar	8	6	4	2
baa	6	12	3	3
car	4	3	8	4
cat	2	3	4	8