Foundations of Machine Learning AI2000 and AI5000

FoML-31
Kernelized Linear Models

Dr. Konda Reddy Mopuri
Department of AI, IIT Hyderabad
July-Nov 2025

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DiL
Data-driven Intelligence & Learning Lab

So far in FoML

- Intro to ML and Probability refresher
- MLE, MAP, and fully Bayesian treatment
- Supervised learning
    a. Linear Regression with basis functions
    b. Bias-Variance Decomposition
    c. Decision Theory & three broad classification strategies
    d. Neural Networks
- Unsupervised learning
    a. K-Means, Hierarchical, GMM for clustering, and PCA

For today

- Equivalent Kernel
- Kernelizing Linear models

Recap

- Bayesian Regression (foml-11)

Equivalent Kernel

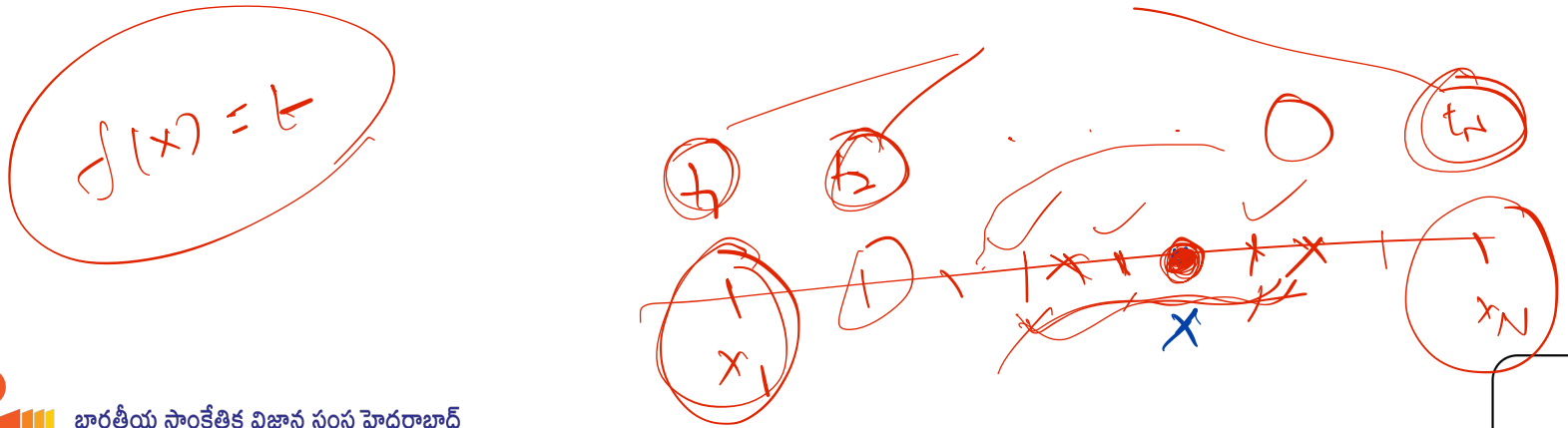
Equivalent Kernel formuluation

- The predictive distribution: $p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$

$$\begin{aligned}\mathbf{m}_N &= \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}.\end{aligned}$$

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$

$K(x, x_n)$



Equivalent Kernel formulation

- The predictive distribution: $p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$

$$\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$
$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}.$$

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$

- predictive mean: $y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} = \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}_n) t_n$

$$k(x, x_n) = \beta \, \phi(x)^{\mathrm{T}} S_N \phi(x_n)$$

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(x, x_n) t_n$$

भारतीय सांकेतिक विज्ञान संस्थ हैदराबाद
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DiL
Data-driven Intelligence & Learning Lab

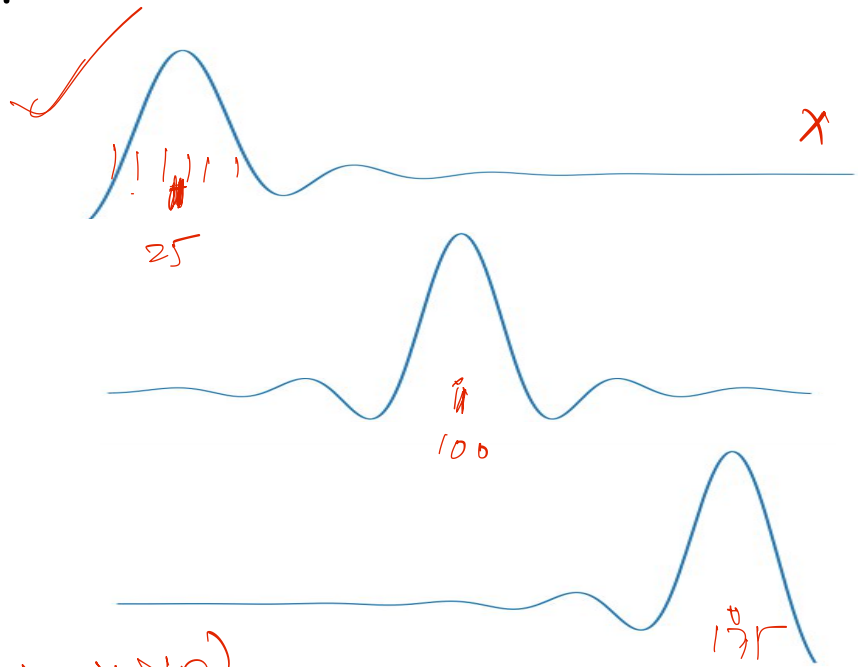
Equivalent Kernel formulation

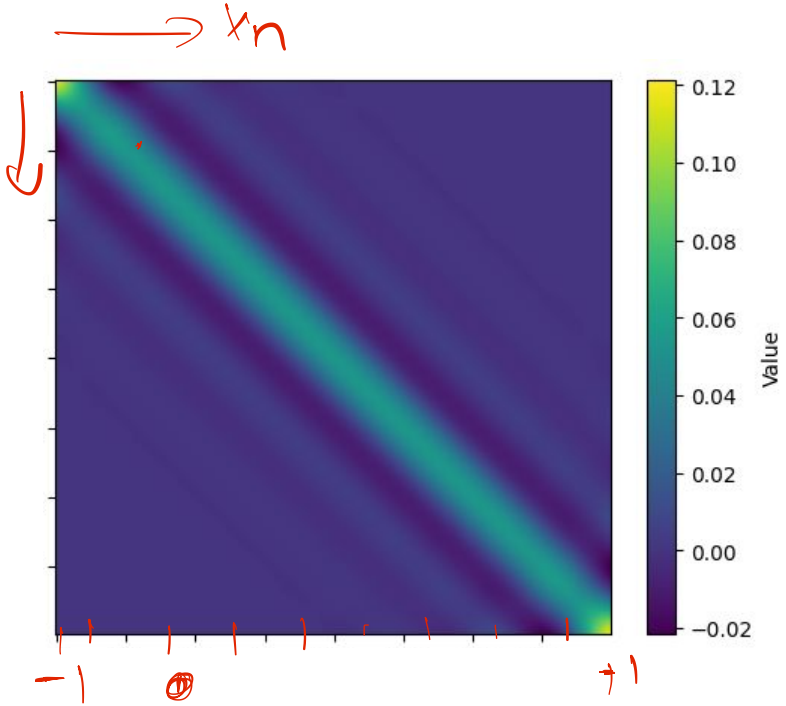
$$k(\mathbf{x}, \mathbf{x}_n) = \beta \boldsymbol{\phi}(\mathbf{x})^T \mathbf{S_N} \boldsymbol{\phi}(\mathbf{x_n})$$

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n$$

- K - smoother matrix or equivalent kernel
  - It depends on all the input samples
- Functions that make predictions by taking linear combinations of the training set target values - linear smoothers

Equivalent kernel for Gaussian Basis functions





Training samples close to x contribute more!

Covariance between two predictions

$$\mathrm{Cov}[t_1, t_2 | \mathbf{x}_1, \mathbf{x}_2] =$$

$$= \mathrm{Cov}_{\mathbf{w}}\left[\phi(x_1)^T \mathbf{w},\ \mathbf{w}^T \phi(x_2)\right]$$

$$= \mathbb{E}_{\mathbf{w}}\left[\phi(x_1)^T \mathbf{w}\mathbf{w}^T \phi(x_2)\right] - \mathbb{E}_{\mathbf{w}}\left[\phi(x_1)^T \mathbf{w}\right] \mathbb{E}_{\mathbf{w}}\left[\mathbf{w}^T \phi(x_2)\right]$$

$$= \phi(x_1)^T \mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right] \phi(x_2) - \phi(x_1)^T \mathbb{E}[\mathbf{w}]\, \mathbb{E}[\mathbf{w}^T]\, \phi(x_2)$$

$$= \phi(x_1)^T \left(\mathbb{E}[\mathbf{w}\mathbf{w}^T] - \mathbb{E}[\mathbf{w}]\, \mathbb{E}[\mathbf{w}^T]\right) \phi_2(x_2)$$

$$= \phi(x_1)^T S_N\, \phi(x_2) = \frac{1}{\beta} k(x_1, x_2)$$

$$=$$

$\mathcal{N}(0, \lambda^{-1} I)$

$\mathcal{N}(m_N, S_N)$

Alternate approach to parametric modeling

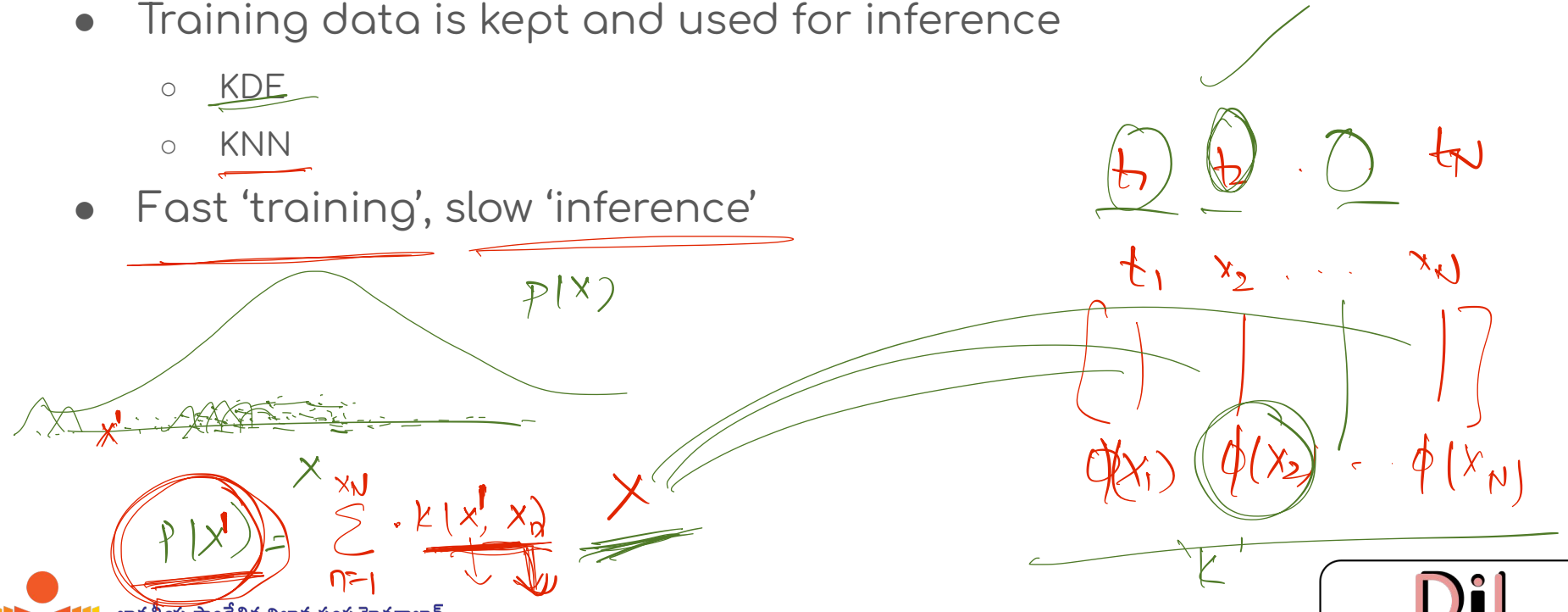
- Instead of working with basis functions
- Define a localized kernel to make predictions for new points
  - Gaussian Processes

Summary - Parametric Models

- Use fixed basis function to project the data
  - Learning: regression, classification
- Learnable basis functions: neural networks
- Training
  - MLE, MAP → point estimate W
  - Full Bayesian → posterior on W
- Test time
  - Don't need the training data
  - Work with W or its distribution

Memory-based Methods

- Training data is kept and used for inference
  - KDE
  - KNN
- Fast 'training', slow 'inference'



Non-parametric Kernel Methods

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DiL
Data-driven Intelligence
& Learning Lab

Non-parametric methods

- Kernel methods
  - Use training data for test time predictions
- 'Dual representation' for the linear parametric models
  - Equivalent kernels

Kernelized Ridge Regression

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

- Solution to w takes a form of linear combination of basis vectors

$$\frac{\partial}{\partial w} J(w) = 0 \qquad \sum_{n=1}^{N} \left(w^T \phi(x_n) - t_n\right)\phi(x_n) + \lambda w = 0$$

$$w = -\frac{1}{\lambda}\sum_{n=1}^{N} \left(w^T \phi(x_n) - t_n\right)\phi(x_n)$$

a_n

$$w = \Phi^T a \qquad (m \times 1) = (m \times N)(N \times 1)$$

Kernelized Ridge Regression

- Instead of working with w, let us work with a

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

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$

w → a

N×1 N×1

Kernelized Ridge Regression

- Instead of working with w, let us work with a

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$$

- Introduce a gram matrix K

$$K_{nm} = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

$K = \Phi\Phi^T$

N×N N×M M×N

Kernelized Ridge Regression

- Optimizing for a

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}.$$

$$\frac{\partial}{\partial a} J(a) \qquad KKa - Kt + \lambda Ka = 0$$

$$Kt = KKa + \lambda Ka$$

$$t = \frac{Ka + \lambda I_N a}{(K + \lambda I_N)^{-1} t}$$

$$a =$$

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{t}.$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{t}$$

Primal and Dual perspectives

$$\mathbf{w} = \Phi^T \mathbf{a}$$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$$

Inference

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}.$$

$$y(\mathbf{x}, \mathbf{a}) = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x})$$

Inferen

Primal and Dual perspectives

$$\mathbf{w} = \Phi^T \mathbf{a}$$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$$

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}.$$

$$y(\mathbf{x}, \mathbf{a}) = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x})$$

- Compute (train)
  - $O(M^3)$ vs. $O(N^3)$
- Compute (inference)
  - O(M) vs. O(NM)

Primal and Dual perspectives

-

Next

- Kernel methods with ‘sparsity’ in solutions