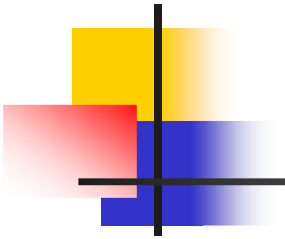
IDEAL 2007 Presentation

Sparse Kernel Modelling: A Unified Approach

S. Chen[†], X. Hong[‡] and C.J. Harris[†]

[†] **School of Electronics and Computer Science,**
University of Southampton, SO17 1BJ, UK

[‡] **Department of Cybernetics,**
University of Reading, RG6 6AY, UK



Outline

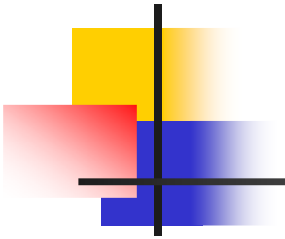
- Motivations and existing approaches for parsimonious **kernel data modelling**

- The proposed unified data modelling approach for
  - ☆ **regression** (supervised learning)
  - ☆ **classification** (supervised learning)
  - ☆ **density estimation** (unsupervised learning)

- Experimental investigation of the proposed approach and comparison with some existing techniques

Motivations

- In kernel data modelling, training data are all one has to build a model
  - Yet objective of modelling from data is not that model simply fits training data well
  - Rather, goodness of a model is characterised by its **generalisation capability**, **interpretability** and ease of **knowledge extraction**
- All depend crucially on ability to construct **parsimonious** models that capture underlying **data generating mechanism**
- How to measure **goodness** of modelling process
  - **Generalisation** performance
  - **Sparsity** level or model size
  - Computational **efficiency** of modelling process



Data Modelling Classes

- **Supervised** learning
  - **Regression**: infer model $\hat{f} : \mathcal{R}^m \rightarrow \mathcal{R}$ that captures data generating machanism $f : \mathcal{R}^m \rightarrow \mathcal{R}$ based on training data $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$ generated from $y = f(\mathbf{x}) + e$, e being observation noise
  - **Classification** (two-class): infer classifier $\hat{f} : \mathcal{R}^m \rightarrow \{-1, +1\}$ that models data generating machanism $f : \mathcal{R}^m \rightarrow \{-1, +1\}$ based on training data $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$, y_k being class label for $\mathbf{x}_k$
- **Unsupervised** learning
  - **Probability density function** estimation: infer estimate $\hat{f} : \mathcal{R}^m \rightarrow \mathcal{R}_+$, based on training data $D_N = \{\mathbf{x}_k\}_{k=1}^N$ drawn from unknown true density $f : \mathcal{R}^m \rightarrow \mathcal{R}_+$
  - Desired response for $\mathbf{x}_k$ is unavailable, and this is **constrained** learning, as $\int_{\mathcal{R}^m} \hat{f}(\mathbf{u})\, d\mathbf{u} = 1$

Overview of Existing Methods

- **Sparse kernel modelling** techniques, e.g. **support vector machines**
  - ☆ From full kernel model, try to obtain sparse representation by making many kernel weights to (near) zeros
  - ☆ Robust and optimal; in practice, not as sparse as OLS approach, and a few hyperparameters to tune
- **Orthogonal-least-squares** algorithm for forward selection,
  - ☆ Use computationally efficient OLS to choose a small subset of significant kernels one by one
  - ☆ Suboptimal; in practice, much sparser models with equally good generalisation performance, and fewer hyperparameters to tune
- This work adopts OLS for forward selection based on **leave-one-out** test criterion and **local regularisation**

Unified Data Modelling

- Placing a **kernel** on each training data $\mathbf{x}_k$ and **linearly** combining all model bases

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^{N} \beta_k K_\rho(\mathbf{x}, \mathbf{x}_k)$$

- Adavantage is linear least squares solution readily available for weights β_k, but it is critically important to obtain **sparse** representation
- Gaussian kernel

$$K_\rho(\mathbf{x}, \mathbf{c}_k) = \begin{cases} e^{-\frac{\|\mathbf{x}-\mathbf{c}_k\|^2}{2\rho^2}}, & \text{for regression and classification,} \\ \frac{1}{(2\pi\rho^2)^{m/2}} e^{-\frac{\|\mathbf{x}-\mathbf{c}_k\|^2}{2\rho^2}}, & \text{for density estimation,} \end{cases}$$

- Kernel width ρ is usually not provided by modelling algorithm itself and must be determined via **cross validation**

Regression Modelling

- At a training point $(\mathbf{x}_k, y_k) \in D_N$, **kernel model** can be expressed as

$$y_k = \hat{y}_k + \epsilon_k = \sum_{i=1}^{N} \beta_i K_\rho(\mathbf{x}_k, \mathbf{x}_i) + \epsilon_k = \boldsymbol{\phi}^T(k)\boldsymbol{\beta} + \epsilon_k$$

  where $\epsilon_k = y_k - \hat{y}_k$ is **modelling error** at $\mathbf{x}_k$, $\boldsymbol{\beta} = [\beta_1 \ \beta_2 \cdots \beta_N]^T$ and $\boldsymbol{\phi}(k) = [K_{k,1} \ K_{k,2} \cdots K_{k,N}]^T$ with $K_{k,i} = K_\rho(\mathbf{x}_k, \mathbf{x}_i)$

- By defining **regression matrix**

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \cdots \boldsymbol{\phi}_N]$$

  with $\boldsymbol{\phi}_k = [K_{1,k} \ K_{2,k} \cdots K_{N,k}]^T$ for $1 \le k \le N$, $\mathbf{y} = [y_1 \ y_2 \cdots y_N]^T$ and $\boldsymbol{\epsilon} = [\epsilon_1 \ \epsilon_2 \cdots \epsilon_N]^T$, **regression model** over D_N can be expressed as

$$\mathbf{y} = \boldsymbol{\Phi}\,\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- Note $\boldsymbol{\phi}_k$ is k-th **column** of $\boldsymbol{\Phi}$, while $\boldsymbol{\phi}^T(k)$ denotes k-th **row** of $\boldsymbol{\Phi}$

Orthogonal Decomposition

- **Orthogonal decomposition** of **regression matrix**: $\mathbf{\Phi} = \mathbf{WA}$, where

$$\mathbf{A} = \begin{bmatrix} 1 & a_{1,2} & \cdots & a_{1,N} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{N-1,N} \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

  $\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \cdots \mathbf{w}_N]$ with **orthogonal columns**: $\mathbf{w}_i^T \mathbf{w}_j = 0$, if $i \neq j$

- **Regression model** can alternatively be expressed as $\mathbf{y} = \mathbf{W}\,\mathbf{g} + \boldsymbol{\epsilon}$, where new weight vector $\mathbf{g} = [g_1 \ g_2 \cdots g_N]^T$ satisfies $\mathbf{A}\,\boldsymbol{\beta} = \mathbf{g}$

- Space spanned by **original bases** $\boldsymbol{\phi}_k$ is identical to space spanned by **orthogonal bases** $\mathbf{w}_k$, and model is equivalently expressed by $\hat{y}_k = \mathbf{w}^T(k)\,\mathbf{g}$, where $\mathbf{w}^T(k) = [w_{k,1} \ w_{k,2} \cdots w_{k,N}]$ is k-th row of $\mathbf{W}$

Local Regularisation

- **Regularised** LS solution for $\mathbf{g}$ is obtained by minimising

$$J_R(\mathbf{g}, \boldsymbol{\lambda}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \sum_{i=1}^{N} \lambda_i g_i^2 = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \mathbf{g}^T \boldsymbol{\Lambda} \mathbf{g}$$

- **Hyperparameters** λ_i specify **prior distributions** of $\mathbf{g}$, and initially λ_i are set to same small value (same flat distribution for each prior of g_i)
- **Evidence procedure** is used to update regularisation parameters

$$\lambda_i^{\text{new}} = \frac{\gamma_i^{\text{old}}}{N - \gamma^{\text{old}}} \frac{\boldsymbol{\epsilon}^T \boldsymbol{\epsilon}}{g_i^2}, \ 1 \leq i \leq N$$

where g_i for $1 \leq i \leq N$ denote current estimated parameter values, and

$$\gamma = \sum_{i=1}^{N} \gamma_i \quad \text{with} \quad \gamma_i = \frac{\mathbf{w}_i^T \mathbf{w}_i}{\lambda_i + \mathbf{w}_i^T \mathbf{w}_i}$$

- A few iterations (typically ≤ 10) are sufficient to find (near) optimal $\boldsymbol{\lambda}$

Leave-One-Out Cross Validation

- **Leave-one-out** cross validation
  - ☆ **Remove** k-th data from D_N and use resultant $D_n \backslash (\mathbf{x}_k, y_k)$ to identify a n-term model, denoting as $\hat{f}^{(n,-k)}$
  - ☆ **Test error** for this n-term model calculated on $(\mathbf{x}_k, y_k)$ not used in training is

$$\epsilon_k^{(n,-k)} = y_k - \hat{f}^{(n,-k)}(\mathbf{x}_k) = y_k - \hat{y}_k^{(n,-k)}$$

  - ☆ **Repeat** for $1 \leq k \leq N$ to obtain leave-one-out test mean square error

$$J_n = \frac{1}{N} \sum_{k=1}^{N} \left(\epsilon_k^{(n,-k)} \right)^2$$

    which is a measure of n-term model's **generalisation** performance

- **No need** to repeatedly remove a data point and identify corresponding model

Leave-One-Out Model Selection

- It can be shown that **leave-one-out** test error is $\epsilon_k^{(n,-k)} = \epsilon_k^{(n)} / \eta_k^{(n)}$
  - ☆ n-term **modelling error** $\epsilon_k^{(n)}$ can be expressed as

$$\epsilon_k^{(n)} = \epsilon_k^{(n-1)} - w_{k,n} g_n$$

    where $w_{k,n}$ is k-th element of $\mathbf{w}_n$
  - ☆ Leave-one-out **error weighting** $\eta_k^{(n)}$

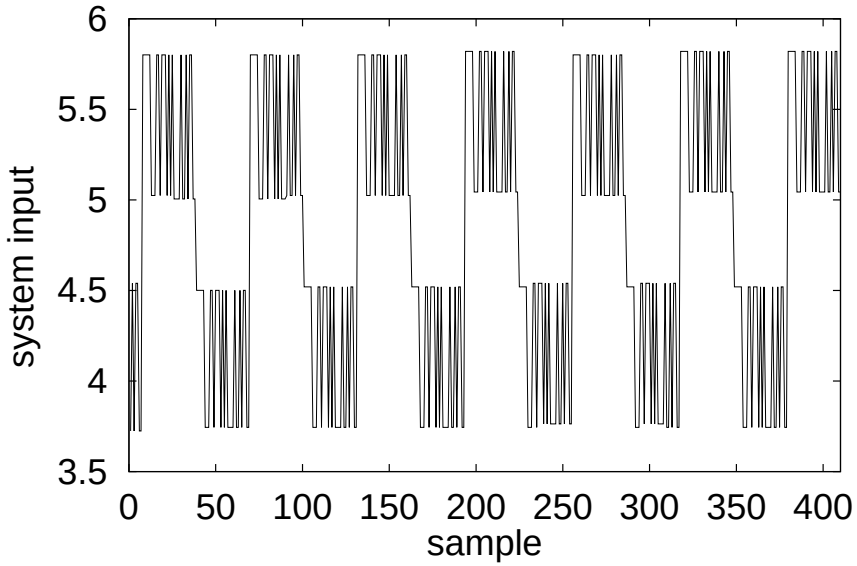
$$\eta_k^{(n)} = \eta_k^{(n-1)} - \frac{w_{k,n}^2}{\mathbf{w}_n^T \mathbf{w}_n + \lambda_n}$$

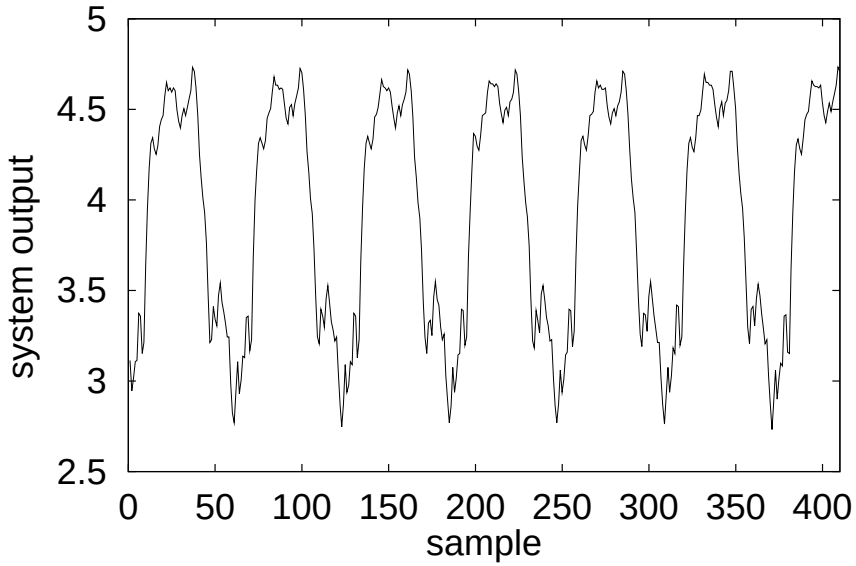
- At n-th stage of **OLS selection** procedure, n-th model term is selected to minimise leave-one-out **test mean square error** J_n
- Selection procedure is **automatically** terminated when $J_{N_s+1} \geq J_{N_s}$, where $N_s \ll N$, yielding N_s-term **sparse model**

Engine Data Set

- Modelling relationship between fuel rack position (input u_k) and engine speed (output y_k) for a **diesel engine** operated at **low engine speed**

  - Data set contained 410 samples with first 210 points for training and last 200 points for test
  - This data set can be represented as $y_k = f(\mathbf{x}_k) + e_k$ where e_k denotes system noise and $\mathbf{x}_k = [y_{k-1}\ u_{k-1}\ u_{k-2}]^T$
  - Optimal Gaussian kernel variance $\rho^2 = 1.69$ was found empirically



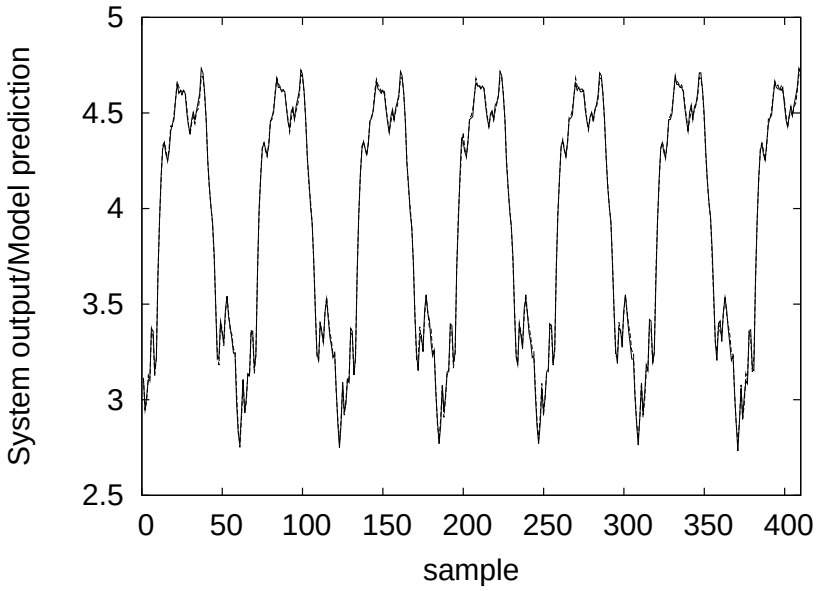


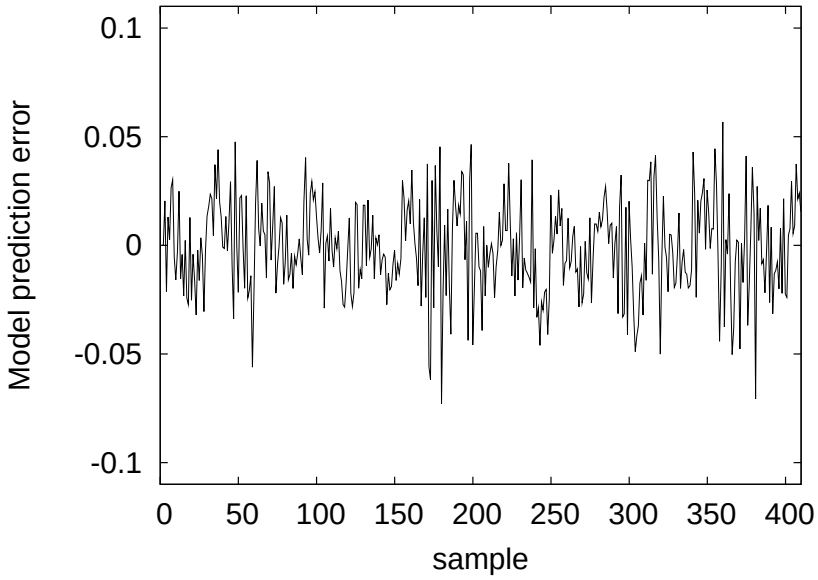
Engine Data Set (continue)

- Modelling accuracy for the engine data set using proposed OLS and SVM algorithms

algorithm	model size	training MSE	test MSE
OLS	22	0.000453	0.000490
SVM	92	0.000447	0.000498

- Modelling for engine data set using OLS: (a) prediction $\hat{y}_k$ (dashed) superimposed on system output y_k (solid), and (b) prediction error $\epsilon_k = y_k - \hat{y}_k$





Boston Housing Data Set

- **Boston housing** data set: a regression benchmark comprised 506 data points with 14 variables
  - Predict median house value from remaining 13 attributes
  - 456 data points were randomly selected for training and remaining 50 data points were used to form test set
  - Average results were given over 100 repetitions
  - Optimal Gaussian kernel width was found via cross validation
- Modelling accuracy for Boston housing data set: Results were averaged over 100 realizations and quoted as mean±standard deviation

algorithm	model size	training MSE	test MSE
OLS	58.6 ± 11.3	12.9690 ± 2.6628	17.4157 ± 4.6670
SVM	243.2 ± 5.3	6.7986 ± 0.4444	23.1750 ± 9.0459

Kernel Classification

- Given training set $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$, where $\mathbf{x}_k \in \mathcal{R}^m$ is **pattern vector** and $y_k \in \{-1, +1\}$ is **class label** for $\mathbf{x}_k \Rightarrow$ construct **kernel classifier**

$$\tilde{y}_k = \text{sgn}\,(\hat{y}_k) \;\text{ with }\; \hat{y}_k = \sum_{i=1}^{N} \beta_i K_\rho(\mathbf{x}_k, \mathbf{x}_i)$$

  $\tilde{y}_k$ is estimated class label for $\mathbf{x}_k$, $\text{sgn}(y) = -1$ if $y \leq 0$ and $\text{sgn}(y) = +1$ if $y > 0$

- Define **modelling error** $\epsilon_k = y_k - \hat{y}_k \Rightarrow$ classification model over D_N can be expressed as: $\mathbf{y} = \mathbf{\Phi}\,\boldsymbol{\beta} + \boldsymbol{\epsilon}$

- Or equivalently in **orthogonal regression model** form: $\mathbf{y} = \mathbf{W}\,\mathbf{g} + \boldsymbol{\epsilon}$, where all relevant notations are as defined for regression modelling

- **Classifier construction** has same **regression modelling** form, but how good a classifier is is judged by its **misclassification rate**

Leave-One-Out Misclassification Rate

- Define leave-one-out **signed decision variable**: $s_k^{(n,-k)} = y_k \hat{y}_k^{(n,-k)}$, where $\hat{y}_k^{(n,-k)}$ is **test output** of n-term model evaluated at k-th data sample not used in training
- Leave-one-out **misclassification rate** can be computed as

$$J_n = \frac{1}{N} \sum_{k=1}^{N} \mathcal{I}_d \left(s_k^{(n,-k)} \right)$$

  indicator function $\mathcal{I}_d(y) = 1$ if $y \leq 0$ and $\mathcal{I}_d(y) = 0$ if $y > 0$

- From leave-one-out n-term modelling error, it can be shown that leave-one-out n-term signed decision variable is: $s_k^{(n,-k)} = \psi_k^{(n)} / \eta_k^{(n)}$
- **Leave-one-out** error weighting $\eta_k^{(n)}$ can be computed recursively and similarly

$$\psi_k^{(n)} = \psi_k^{(n-1)} + y_k g_n w_{k,n} - \frac{w_{k,n}^2}{\mathbf{w}_n^T \mathbf{w}_n + \lambda_n}$$

Breast Cancer Data Set

Average classification test error rate in % over 100 realizations

method	test error rate	model size
RBF-Network	27.64 ± 4.71	5
AdaBoost with RBF-Network	30.36 ± 4.73	5
LP-Reg-AdaBoost (-"-)	26.79 ± 6.08	5
QP-Reg-AdaBoost (-"-)	25.91 ± 4.61	5
AdaBoost-Reg (-"-)	26.51 ± 4.47	5
SVM with RBF-Kernel	26.04 ± 4.74	not available
Kernel Fisher Discriminant	24.77 ± 4.63	200
OLS	25.74 ± 5.00	6.0 ± 2.0

Data and first 7 results from:

`http://ida.first.fhg.de/projects/bench/benchmarks.htm`

Diabetis Data Set

Average classification test error rate in % over 100 realizations

method	test error rate	model size
RBF-Network	24.29 ± 1.88	15
AdaBoost with RBF-Network	26.47 ± 2.29	15
LP-Reg-AdaBoost (-"-)	24.11 ± 1.90	15
QP-Reg-AdaBoost (-"-)	25.39 ± 2.20	15
AdaBoost-Reg (-"-)	23.79 ± 1.80	15
SVM with RBF-Kernel	23.53 ± 1.73	not available
Kernel Fisher Discriminant	23.21 ± 1.63	468
OLS	23.00 ± 1.70	6.0 ± 1.0

Data and first 7 results from:

`http://ida.first.fhg.de/projects/bench/benchmarks.htm`

Kernel Density Estimation

- **Parzen window estimate** $\hat{f}(\mathbf{x}; \boldsymbol{\beta}_{\text{Par}}, \rho_{\text{Par}})$ can be regarded as "observation" of **true density** contaminated by "observation noise"

$$\hat{f}(\mathbf{x}; \boldsymbol{\beta}_{\text{Par}}, \rho_{\text{Par}}) = f(\mathbf{x}) + \tilde{\epsilon}(\mathbf{x})$$

- **Kernel density estimation** can be viewed as **constrained regression** with Parzen window estimate as **desired response**

$$\hat{f}(\mathbf{x}; \boldsymbol{\beta}_{\text{Par}}, \rho_{\text{Par}}) = \sum_{k=1}^{N} \beta_k K_\rho(\mathbf{x}, \mathbf{x}_k) + \epsilon(\mathbf{x})$$

  subject to constraints $\beta_k \geq 0$, $1 \leq k \leq N$, and $\boldsymbol{\beta}^T \mathbf{1}_N = 1$

- Define $y_k = \hat{f}(\mathbf{x}_k; \boldsymbol{\beta}_{\text{Par}}, \rho_{\text{Par}})$ and $\epsilon_k = \epsilon(\mathbf{x}_k) \Rightarrow$ **density estimation** is expressed as **regression modelling**: $\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, or alternatively: $\mathbf{y} = \mathbf{W}\,\mathbf{g} + \boldsymbol{\epsilon}$

- Subject to **nonnegative** and **unity** constraints

Kernel Density Construction

- OLS **sparse kernel regression modelling** algorithm to select sparse N_s-term **subset model**, where $N_s \ll N$
  - This determines structure of density estimate, containing N_s significant kernels
- **Multiplicative nonnegative quadratic programming** to calculate kernel weights
  - Formally, task is to find $\boldsymbol{\beta}_{N_s}$ for regression model

$$\mathbf{y} = \boldsymbol{\Phi}_{N_s} \boldsymbol{\beta}_{N_s} + \boldsymbol{\epsilon}$$

  - Subject to **nonnegative constraint**

$$\beta_i \geq 0, \ 1 \leq i \leq N_s$$

and **unity constraint**

$$\boldsymbol{\beta}_{N_s}^T \mathbf{1}_{N_s} = 1$$

One-Dimensional Example

- Density to be estimated was mixture of Gaussian and Laplacian

$$p(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} + \frac{0.7}{4} e^{-0.7|x+2|}$$

  - Number of training data points was $N = 100$, separate test data set of $N_{\text{test}} = 10,000$ samples was used to calculate L_1 test error

$$L_1 = \frac{1}{N_{\text{test}}} \sum_{k=1}^{N_{\text{test}}} |p(\mathbf{x}_k) - \hat{p}(\mathbf{x}_k; \boldsymbol{\beta}, \rho)|$$

    together with Kullback-Leibler divergence

$$\text{KLD} = \int_{\mathcal{R}^m} p(\mathbf{x}) \log \left(\frac{p(\mathbf{x})}{\hat{p}(\mathbf{x}; \boldsymbol{\beta}, \rho)} \right) d\mathbf{x}$$

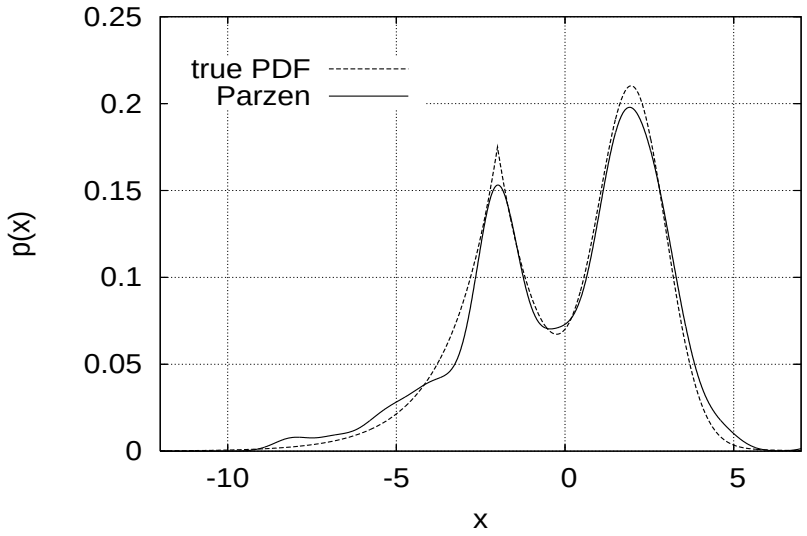
  - Experiment was repeated $N_{\text{run}} = 100$ times, optimal kernel widths were found to be $\rho = 0.54$ and $\rho = 1.1$ empirically for Parzen window estimate and proposed sparse kernel density estimate, respectively

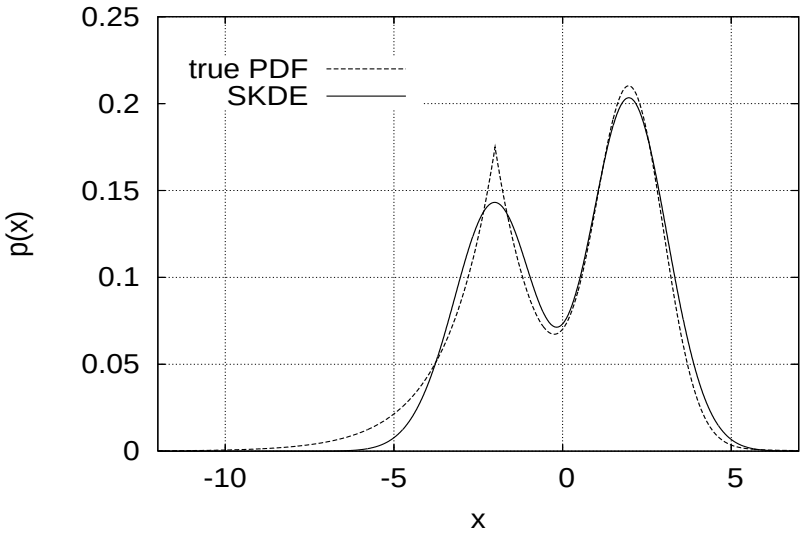
One-Dimensional Example (continue)

- Performance comparsion

method	L_1 test error	K-L divergence	kernel no.
PWE	$(1.9963 \pm 0.6179) \times 10^{-2}$	$(8.0003 \pm 5.1662) \times 10^{-2}$	100 ± 0
OLS	$(2.0213 \pm 0.6535) \times 10^{-2}$	$(8.1419 \pm 5.0102) \times 10^{-2}$	5.1 ± 1.2

- A Parzen window estimate and a sparse kernel density estimate, in comparison with true density





Six-Dimensional Example

- Density to be estimated was mixture of three Gaussian distributions

$$p(\mathbf{x}) = \frac{1}{3}\sum_{i=1}^{3} \frac{1}{(2\pi)^{6/2}} \frac{1}{\det^{1/2}|\mathbf{\Gamma}_i|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \mathbf{\Gamma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}$$

$$\boldsymbol{\mu}_1 = [1.0\ 1.0\ 1.0\ 1.0\ 1.0\ 1.0]^T \quad \mathbf{\Gamma}_1 = \mathrm{diag}\{1.0, 2.0, 1.0, 2.0, 1.0, 2.0\}$$

$$\boldsymbol{\mu}_2 = [-1.0\ -1.0\ -1.0\ -1.0\ -1.0\ -1.0]^T \quad \mathbf{\Gamma}_2 = \mathrm{diag}\{2.0, 1.0, 2.0, 1.0, 2.0, 1.0\}$$

$$\boldsymbol{\mu}_3 = [0.0\ 0.0\ 0.0\ 0.0\ 0.0\ 0.0]^T \quad \mathbf{\Gamma}_3 = \mathrm{diag}\{2.0, 1.0, 2.0, 1.0, 2.0, 1.0\}$$

  - $N = 600$, $N_{\text{test}} = 10,000$ and $N_{\text{run}} = 100$

- Performance comparsion

method	L_1 test error	kernel number
Parzen window estimate	$(3.5195 \pm 0.1616) \times 10^{-5}$	600 ± 0
proposed SKD estimate	$(3.1134 \pm 0.5335) \times 10^{-5}$	9.4 ± 1.9

Conclusions

- A **unified regression framework** has been proposed
  - applicable to supervised **regression** and **classification** problems
  - as well as unsupervised **probability density function learning**
- An efficient algorithm has been developed based on
  - **orthogonal least squares** forward selection procedure
  - incrementally minimises **leave-one-out** criterion coupled with **local regularisation**
  - **multiplicative nonnegative quadratic programming** for kernel density weights
- Proposed method is **computationally efficient**
  - capable of constructing very sparse kernel models with excellent **generalisation capability**