Bachelor Thesis Presentation

Probabilistic Machine Learning Tools For Reduced Order Basis Methods

Lukas Köstler

February 12, 2016

Section 1

Theory

Problem description

- Given: inputs $x_i \in \mathbb{R}^{d_x}$, $i = 1 \ldots N$ and outputs $y_i \in \mathbb{R}^{d_y}$ of some unknown function $f : x \to y$
- Goal: Find a surrogate model which predicts $y = f(x)$ for a new x
- Major Assumption: The output data y_i lies on a lower-dimensional manifold

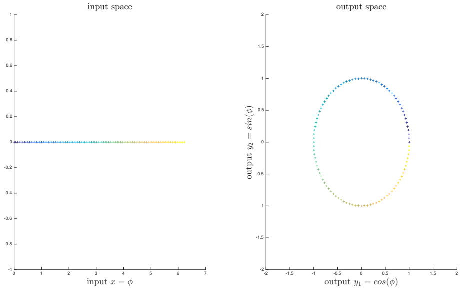


Figure : One example of in- and output data

Basic Idea

- Deal with the non-linearity of the manifold by locally approximating it by affine/linear sub-spaces (i.e. a reduced order basis)
- Associate each data point with the corresponding sub-space
- Learn a rule for the input space, which associates a new point with the "best" subspace

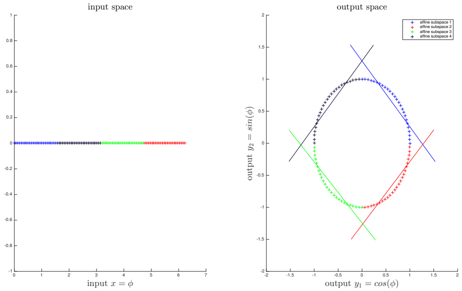


Figure : 4 sub-spaces and the resulting classification of the data points

Probabilistic Formulation

- The model considered can be thought of as a mixture model in the output space with the mixing coefficients depending on x
- $P(y|x) = \sum_{m=1}^{M} \underbrace{P(c=m|x)}_{\text{mixing coefficient for component m}} \overbrace{P(y|c=m)}^{\text{distribution for component m}}$
- To fit the model to data, i.e. train, we first parametrize it (parameters θ)
- Fitting the model is done by maximizing the complete data likelihood (or posterior) using the Expectation Maximization (EM) algorithm

Probabilistic Formulation 2

Complete data log posterior: $\log P(\theta|c, y; x) =$

$$\sum_{n=1}^{N}\sum_{m=1}^{M} 1(c_n = m)\left[\log P(y_n|c_n = m, \theta) \log P(c_n = m|\theta; x_n)\right] + logP(\theta|x) + C \quad (1)$$

For the expectation of the log posterior w.r.t. the distribution of c given some fixed values θ^* of the parameters this yields:

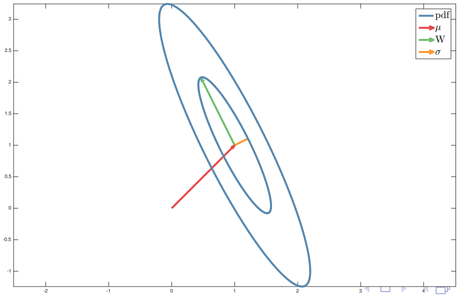
$$Q(\theta|\theta^*) = E_c\left[\log P(\theta|c, y; x)|y; \theta^*\right] \quad (2)$$

$$= Q(\theta_y|\theta^*) + Q(\theta_c|\theta^*) \quad (3)$$

θ_y are the parameters of $P(y|c = m)$ and θ_c of $P(c = m|x)$

Probabilistic Principal Component Analysis

- Probabilistic extension of the well known Principal Component Analysis (PCA)
- PCA finds the q dimensional subspace with the least squared projection error
- Sub-space is represented by the middle μ and the q vectors in $W \in \mathbb{R}^{d_y, q}$
- $P(y|c = m) = \mathcal{N}(\mu_m, \Sigma_m)$ and $\Sigma_m = \sigma_m^2 I + W_m W_m^T$



Obstacles in the way

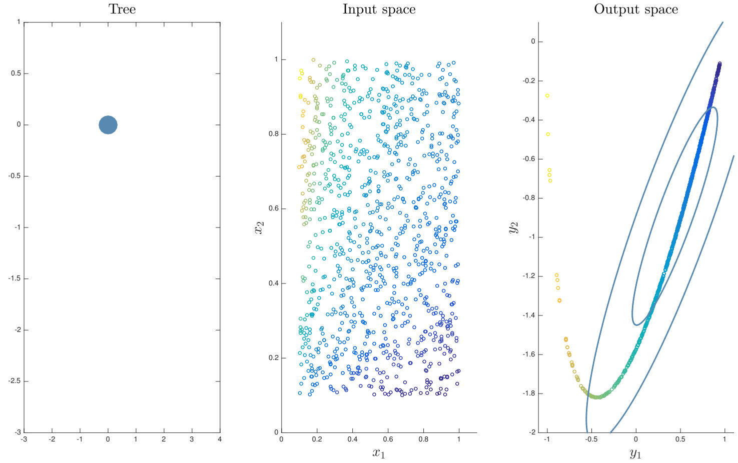
1. The number of mixture components M is hard to determine a priori
2. Finding initial values for the parameters θ is hard (local minima)
3. Multi-class Classification (for $M > 3$) is not easy

Proposed Algorithmic solution

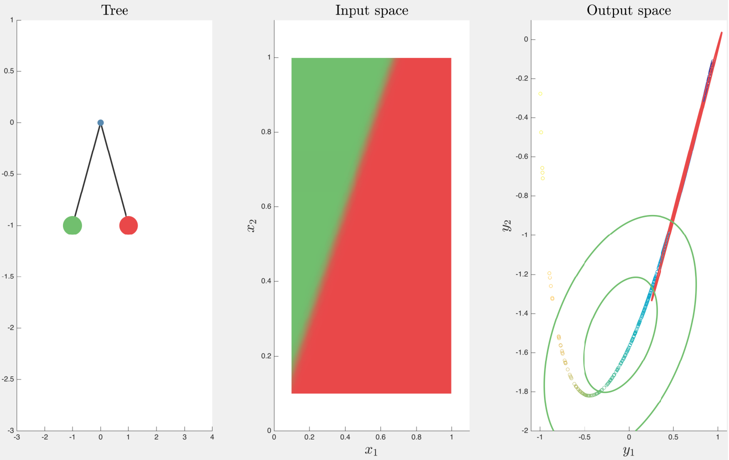
- Start with only one mixing component and iteratively refine the model by adding new components
- The "worst" mixing component is replaced by two new components
- Each point that "belonged" to the original component is "assigned" to one of the two succeeding components
- This leads to a binary tree structure with mixture components on all terminal leafs and binary classification at each internal node

Illustrating Example: Initial configuration

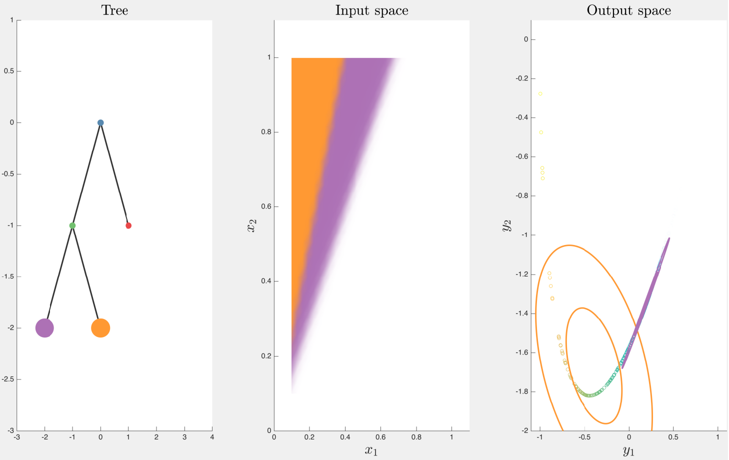
- $(y_1, y_2) = f(x_1, x_2) = \left(cos\left(\sqrt{\frac{x_2}{x_1}}\right), -\sqrt{\frac{x_2}{x_1}} sin\left(\sqrt{\frac{x_2}{x_1}}\right)\right)$



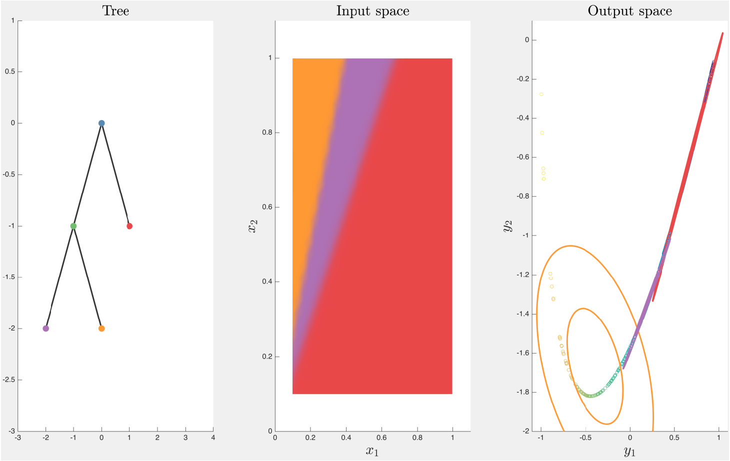
Illustrating Example: After first split



Illustrating Example: Upper left leaf after split



Illustrating Example: Whole tree after two splits



Obstacles no longer in the way

1. The iterative refinement stops at a prescribed level of accuracy
2. At each split initial values for only two PPCAs have to be found. This could be done via e.g. k-means or the responsibility split
3. At each Split only a binary classification problem has to be solved

Classification

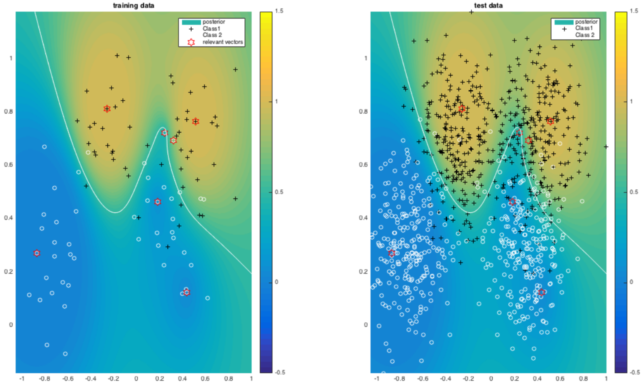
- Because of the Tree structure only a binary classifier is needed
- Tipping: Relevance Vector Machine (RVM) is a probabilistic and mostly sparser version of the support vector machine (SVM)
- Linking function: $P\left(class = 1|x\right) = \sigma\left(w^T \phi\left(x\right)\right) = \frac{1}{1+exp\{-w^T\phi(x)\}}$
- $\phi\left(x\right) = \left(\phi_1\left(x\right), \ldots, \phi_N\left(x\right)\right)$ are called the basis functions
- Class labels: $c_i = 1$ if point i belongs to class 1 and $c_i = 0$ otherwise
- Bernoulli-likelihood:
  $L = P\left(c|w;x\right) = \prod_{i=1}^{N} \sigma\left(w^T \phi\left(x_i\right)\right)^{c_i} \left[1 - \sigma\left(w^T \phi\left(x_i\right)\right)\right]^{1-c_i}$

Basis functions

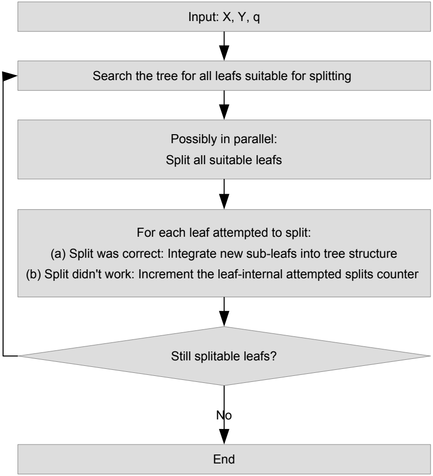
- bias term: $\phi(x) = 1$
- linear: $\phi(x) = x$
- polynomial: e.g. $\phi(x) = x_1 x_2$
- Kernels: $\phi(x) = K\left(x, x^{(j)}\right)$ and $x^{(j)}$ is mostly another data point
  - linear: $K\left(x, x^{(j)}\right) = x^T x^{(j)}$
  - Polynomial: $K\left(x, x^{(j)}\right) = \left(\gamma x^T x^{(j)} + c\right)^d$
  - Gaussian/RBF: $K\left(x, x^{(j)}\right) = exp\left(\frac{-1}{r^2} \left\|x - x^{(j)}\right\|_2^2\right)$
- $K\left(x, x^{(j)}\right)$ represents a dot product in a possibly infinite dimensional feature space

RVM: example: Ripley Synthetic Data

- Gaussian Kernels centered at each data point as basis functions
- Color = class posterior
- White line = decision boundary i.e.
  $P\left(class = 1|x\right) = P\left(class = 2|x\right) = 0.5$



Algorithm: Overview



MCR bound

- The experiments show that the Missclassification Rate (MCR) of each split is a parameter very well describing the overall performance of the algorithm
- Goal: establish a formula for the maximum MCR s.t. the split does decrease the Predicted Squared Error
- Assume $y \sim p\mathcal{N}\left(\mu_1, \sigma_1^2 I + W_1 W_1^T\right) + (1-p)\mathcal{N}\left(\mu_2, \sigma_2^2 I + W_2 W_2^T\right)$
- Formula for the simplified case $q = q_1 = q_2 = 1$, $\|w_1\|_2^2 = \|w_2\|_2^2 = \lambda^2$ and the σ belonging to the PPCA for all points one has:

$$\mathrm{MCR}_{max} = (n-1)\frac{\sigma^2 - \sigma_1^2}{\|\mu_1 - \mu_2\|_2^2 + (1 - cos^2(\angle(w_1, w_2)))\lambda^2}$$

Sharpness-Increase

- Problem: Sometimes the classifier assigns probabilities close to 0.5 to most of the points $\Rightarrow$ the difficulty of the problem is not (notably) decreased by the split
- Solution: After training increase the magnitude of w
- Multiply the likelihood by a heuristic
- Let $a_i = \sigma\left(w^T \phi\left(x_i\right)\right)$, then $\tilde{L} = P\left(c|w;x\right)h\left(w\right)$
- similar to the inverse of the Gini impurity define: $h\left(w\right) = \prod_{i=1}^{N}\left[\frac{1}{\left(a_i(1-a_i)\right)^{\lambda}}\right]^{R_i}, \ \lambda \geq 0$
- I showed that for $w_{new} = \alpha w_{old}$ and $\lambda < \lambda_{max} = f\left(x, R, \frac{w_{old}}{\|w_{old}\|_2}\right)$ the maximizer $\alpha*$ is existent and unique
- For $\lambda = 0.9\lambda_{max}$: $\alpha*$ is about 2 to 10 depending on the problem

Adapted Gaussian Kernels

- For this application it would be best to have:
  $K(x, x_j) = exp\left(\frac{-1}{r^2} \|f(x) - f(x_j)\|_2^2\right)$
- Taylor expansion around x_j: $f(x) \approx f(x_j) + Df(x_j)(x - x_j)$
- $K(x, x_j) \approx exp\left(-\frac{1}{r^2}(x - x_j)^T (Df(x_j))^T Df(x_j)(x - x_j)\right)$
- Example: Let $f(x) = b^T x$ then $K(x, x_j) = exp\left(-\frac{1}{r^2}\left(b^T (x - x_j)\right)^2\right)$
- Challenge: Estimate $Df(x_j)$ in an appropriate way, e.g. when $f(x) \in \{0, 1\}$
- Some more details in my thesis, but still more work to do

Section 2

Examples and Discussion

Kraichnan Orszag three mode problem (KO-3)

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}} z_1 &= z_1 z_3 \\ \frac{\mathrm{d}}{\mathrm{dt}} z_2 &= -z_2 z_3 \\ \frac{\mathrm{d}}{\mathrm{dt}} z_3 &= -z_1^2 + z_2^2\end{aligned} \tag{4}$$

Initial Conditions:

- $z_1(0) = 1$ (fixed)
- $z_3(0) = 1$ (fixed)
- $z_2(0) \in [-0.04, 0.04]$ s.t. $z_2(0) = 0.08x_1 - 0.04$
- $T \in [10, 12]$ s.t. $T = 2x_2 + 10$

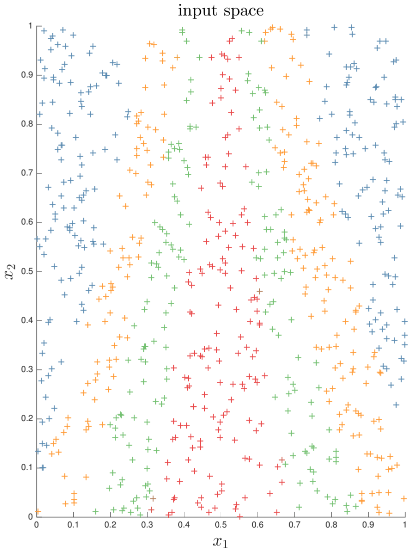
Problem setup:

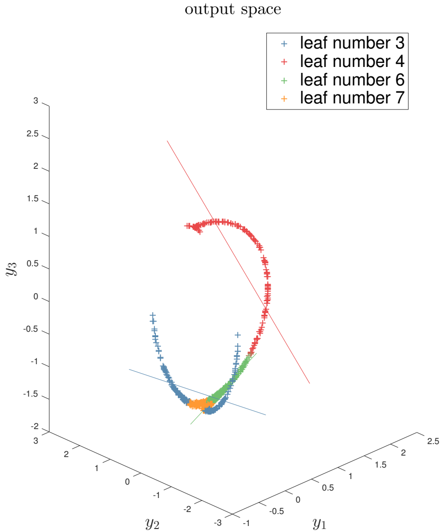
- input: $x_1, x_2 \sim unif(0, 1)$
- output: $y_i = z_i(T)$

KO-3: Parameters

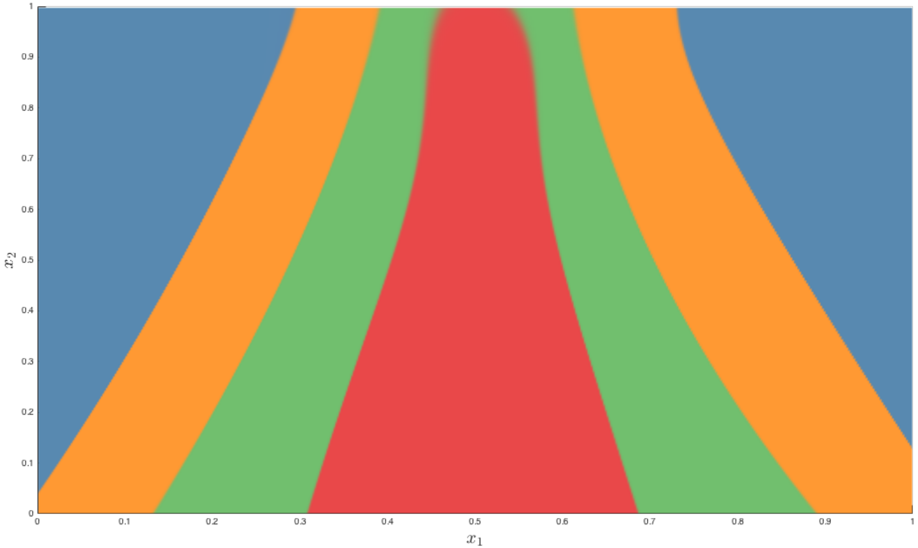
- Parameters used: $N = 50 to 750$ and 5000, $q = 1$, and max. depth $=$ 3, 7 and 11
- Different σ (0.005 and 0.00025) were used and I cross validated with different data-sets
- A very detailed analysis can be found in the thesis
- I used radial basis functions, i.e.: $\phi(x) = (1, K_G(x, x^{(1)}), ..., K_G(x, x^{(m)}))$
- $m = min\,(N, 500)$ and the centers $x^{(1)}, ..., x^{(m)}$ are randomly drawn without replacement from all training data points in each split

KO-3: Results: training points

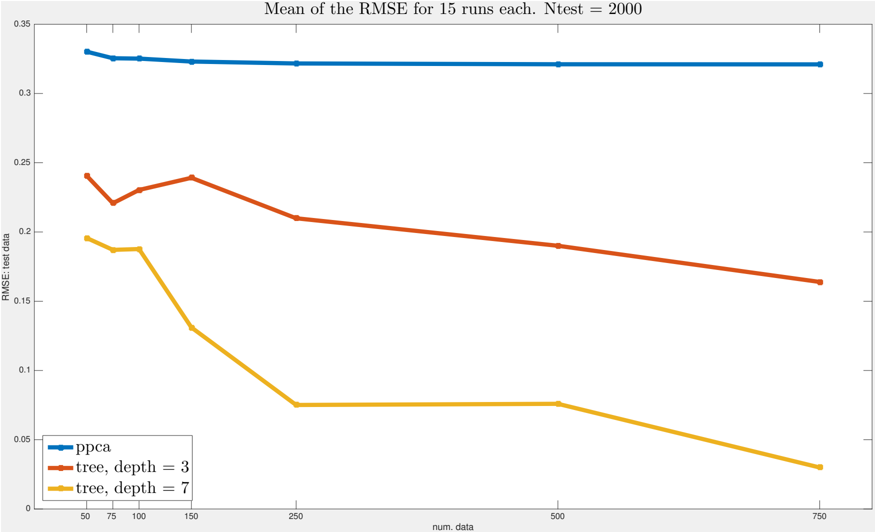




KO-3: Results: segmentation



KO-3: Results: small trees



KO-3: Results: large tree

- max depth $= 11 \rightarrow$ max number nodes $= 2047$
- Tree grew 503 nodes
- Root Mean Squared Test Error ($N_{Test} = 10000$)
  - Tree: $RMSE_{Test} \approx 0.025 \approx 0.041 m_Y$
  - PPCA: $RMSE_{Test} \approx 0.327 \approx 0.542 m_Y$
- $m_Y = \frac{1}{N d_y} \sum_{n=1}^{N} \sum_{d=1}^{d_y} abs\left(y_i^{(n)}\right) \approx 0.604$
- For max depth $= 7$ and $N = 750$ one already achieves $RMSE_{Test} \approx 0.030$

Heat Conduction in 2-D Plate: Setup

- Steady state temperature distribution in a two dimensional plate
- The plate is discretized using $10 \times 10 = 100$ elements
- The temperatures along each of the four boundaries is constant
- The conductivity of each element is chosen at random
- Finite Element Solver written by Constantin

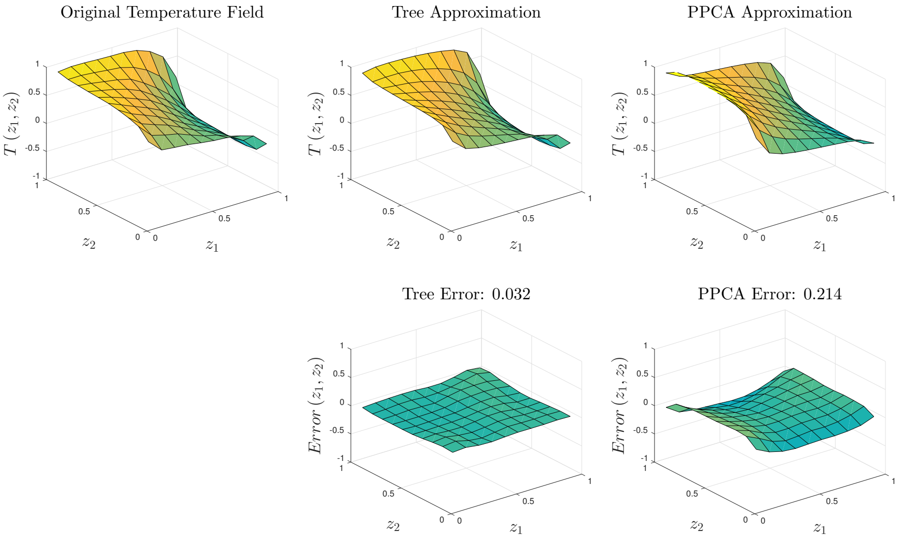
Problem setup:

- input: $x = (T_{lower}, T_{right}, T_{upper}, T_{left}, C_1, \ldots, C_{100})$
- $T_{\ldots} \sim unif(-1, 1)$ and $C_i \sim max(\mathcal{N}(1, 0.4), 0.1)$
- output: $y_i = T_i$ the temperatures of the solution at the element midpoints

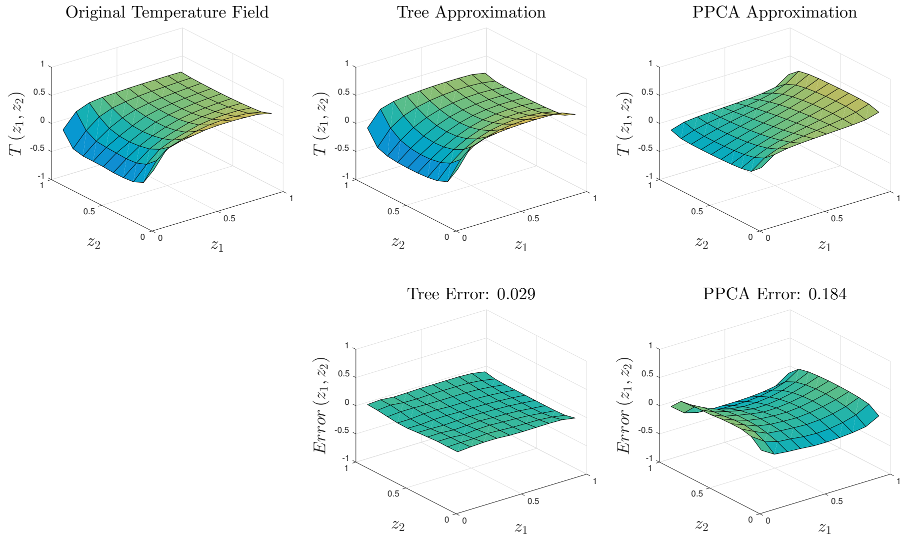
Heat Conduction in 2-D Plate: Result

- Parameters used: $N = 20000$, $q = 2$, $\sigma_{max} = 0.0025 \approx 0.01\sigma_{PPCA}$ and max. depth $= 9$
- I used linear basis functions, i.e.: $\phi(x) = (1, x)$
- Tree grew to full size possible with max. depth $= 9 \rightarrow 255$ internal nodes and 256 leafs
- Root Mean Squared Test Error ($N_{Test} = 10000$)
  - Tree: $RMSE_{Test} \approx 0.035 \approx 0.100 m_Y$
  - PPCA: $RMSE_{Test} \approx 0.236 \approx 0.680 m_Y$
- $m_Y = \frac{1}{Nd_y} \sum_{n=1}^{N} \sum_{d=1}^{d_y} abs\left(y_i^{(n)}\right)$

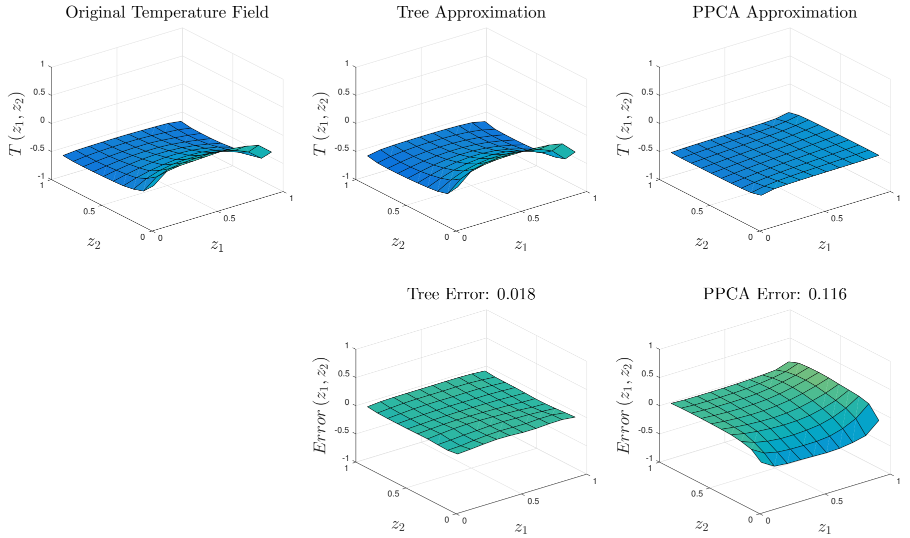
Heat Conduction in 2-D Plate: Some Examples 1



Heat Conduction in 2-D Plate: Some Examples 2



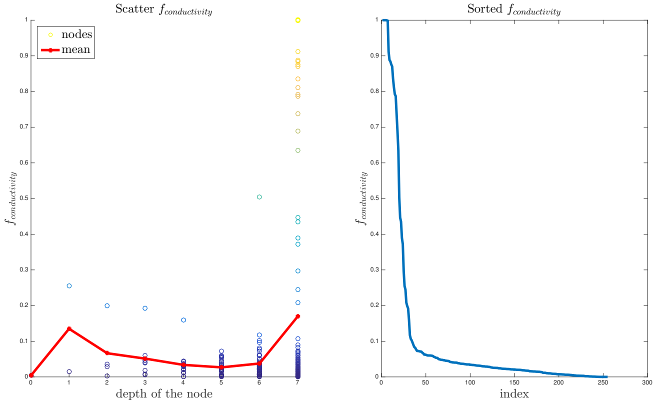
Heat Conduction in 2-D Plate: Some Examples 3



Heat Conduction in 2-D Plate: Discussion

- Fraction of weights acting on the conductivity:

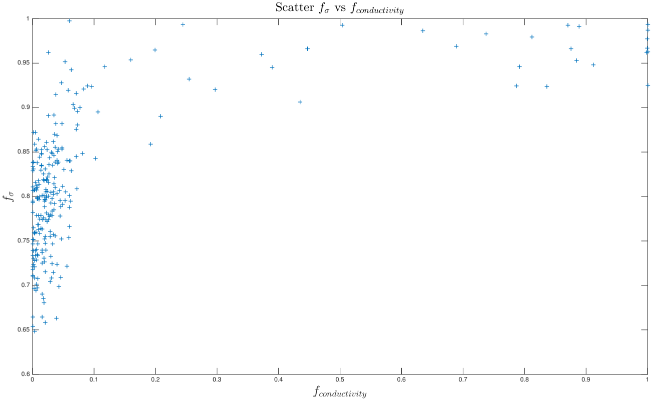
$$f_{conductivity} = \frac{\sum_{i=6}^{105} w_i}{2\sum_{i=2}^{5} w_i + \sum_{i=6}^{105} w_i}$$



Heat Conduction in 2-D Plate: Discussion

- Relative improvement in standard deviation:

$$f_\sigma = \frac{\text{weighted mean}(\sigma_{left}, \sigma_{right})}{\sigma_{before}}$$



Section 3

Results

Advantages of the Algorithm

- The algorithm deals well with high dimensional output in terms of needed data points
- The algorithm deals well with discontinuities
- For low dimensional input non-linearity is well handled
- (P)PCA is well understood and easily interpretable
- The algorithm is fully probabilistic
- Maybe the method represents some generic principle that is applicable to a wide range of problems

Main Challenge

- The main challenge is finding the basis functions / classifier that well suits the structure of the unknown function f
- Possible solution could be:
  - Develop basis functions for common problems (FEM etc.)
  - Try to find basis functions that adapt to the data (Adapted Gaussian Kernels)
  - Rigorously analyze the general structure of the problem and find structure I did not think of

Possible Future Steps

- Tackle the challenge from the slide before
- Use classifiers that optimize some impurity/entropy criterion
- The algorithm is inherently parallel $\rightarrow$ implement a parallel version that can handle large data sets and large (possibly sparse) trees $\Rightarrow$ possibility to compute high dimensional examples with complex basis functions