# Some thoughts about Gaussian Processes

## Model Selection and Large Scale

Olivier Chapelle

Max Planck Institute for Bioligical Cybernetics

Nips Workshop on Gaussian Processes, 2005


# Outline

# Outline

# Model selection criteria

Let $D = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}$ a training set.

## 3 different model selection criteria

Negative log evidence (NLE):

$$-\log P(D) = -\log \int P(D|f)P(f)df.$$

Negative log predictive leave-one-out (NLP-LOO):

$$\sum -\log P((\mathbf{x}_i, y_i)|D \setminus (\mathbf{x}_i, y_i)).$$

Mean squared error leave-one-out (MSE-LOO): $\sum (y_i - f_i(\mathbf{x}_i))^2$

[$f_i$ is the function learned without $\mathbf{x}_i$].

# Model selection criteria

Let $D = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}$ a training set.

## 3 different model selection criteria

Negative log evidence (NLE):

$$-\log P(D) = -\log \int P(D|f)P(f)df.$$

Negative log predictive leave-one-out (NLP-LOO):

$$\sum -\log P((\mathbf{x}_i, y_i)|D \setminus (\mathbf{x}_i, y_i)).$$

Mean squared error leave-one-out (MSE-LOO): $\sum (y_i - f_i(\mathbf{x}_i))^2$

[$f_i$ is the function learned without $\mathbf{x}_i$].

# Model selection criteria

Let $D = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}$ a training set.

## 3 different model selection criteria

Negative log evidence (NLE):

$$-\log P(D) = -\log \int P(D|f)P(f)df.$$

Negative log predictive leave-one-out (NLP-LOO):

$$\sum -\log P((\mathbf{x}_i, y_i)|D \setminus (\mathbf{x}_i, y_i)).$$

Mean squared error leave-one-out (MSE-LOO): $\sum (y_i - f_i(\mathbf{x}_i))^2$

[$f_i$ is the function learned without $\mathbf{x}_i$].

# Connections

## Link NLE – NLP-LOO

NLE $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j>i})$.

NLP-LOO $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j\neq i})$.

$\longrightarrow$ The NLP-LOO conditions more on the data.

In other words, the prior has more influence in the NLE.

## Link NLP-LOO – MSE-LOO

MSE-LOO $\quad \sum (y_i - f_i(\mathbf{x}_i))^2$.

NLP-LOO $\quad \sum \frac{(y_i - f_i(\mathbf{x}_i))^2}{2v_i(\mathbf{x}_i)} + \log(v_i(\mathbf{x}_i))$, where $v_i$ is the predictive variance computed without $\mathbf{x}_i$.

$\longrightarrow$ Predictive variances are ignored in MSE-LOO.

The best criterion is problem and objective dependent.

# Connections

## Link NLE – NLP-LOO

NLE $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j>i})$.

NLP-LOO $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j\neq i})$.

$\longrightarrow$ The NLP-LOO conditions more on the data.

In other words, the prior has more influence in the NLE.

## Link NLP-LOO – MSE-LOO

MSE-LOO $\quad \sum (y_i - f_i(\mathbf{x}_i))^2$.

NLP-LOO $\quad \sum \frac{(y_i - f_i(\mathbf{x}_i))^2}{2v_i(\mathbf{x}_i)} + \log(v_i(\mathbf{x}_i))$, where $v_i$ is the predictive variance computed without $\mathbf{x}_i$.

$\longrightarrow$ Predictive variances are ignored in MSE-LOO.

The best criterion is problem and objective dependent.

# Connections

## Link NLE – NLP-LOO

NLE $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j>i})$.

NLP-LOO $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j\neq i})$.

$\longrightarrow$ The NLP-LOO conditions more on the data.

In other words, the prior has more influence in the NLE.

## Link NLP-LOO – MSE-LOO

MSE-LOO $\quad \sum (y_i - f_i(\mathbf{x}_i))^2$.

NLP-LOO $\quad \sum \frac{(y_i - f_i(\mathbf{x}_i))^2}{2v_i(\mathbf{x}_i)} + \log(v_i(\mathbf{x}_i))$, where $v_i$ is the predictive variance computed without $\mathbf{x}_i$.

$\longrightarrow$ Predictive variances are ignored in MSE-LOO.

The best criterion is problem and objective dependent.

# Connections

## Link NLE – NLP-LOO

NLE $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j>i})$.

NLP-LOO $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j\neq i})$.

$\longrightarrow$ The NLP-LOO conditions more on the data.

In other words, the prior has more influence in the NLE.

## Link NLP-LOO – MSE-LOO

MSE-LOO $\quad \sum (y_i - f_i(\mathbf{x}_i))^2$.

NLP-LOO $\quad \sum \frac{(y_i - f_i(\mathbf{x}_i))^2}{2v_i(\mathbf{x}_i)} + \log(v_i(\mathbf{x}_i))$, where $v_i$ is the predictive variance computed without $\mathbf{x}_i$.

$\longrightarrow$ Predictive variances are ignored in MSE-LOO.

The best criterion is problem and objective dependent.

# Connections

## Link NLE – NLP-LOO

NLE $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j>i})$.

NLP-LOO $\quad \sum -\log P((\mathbf{x}_i, y_i)|\{(\mathbf{x}_j, y_j)\}_{j\neq i})$.

$\longrightarrow$ The NLP-LOO conditions more on the data.

In other words, the prior has more influence in the NLE.

## Link NLP-LOO – MSE-LOO

MSE-LOO $\quad \sum (y_i - f_i(\mathbf{x}_i))^2$.

NLP-LOO $\quad \sum \frac{(y_i - f_i(\mathbf{x}_i))^2}{2v_i(\mathbf{x}_i)} + \log(v_i(\mathbf{x}_i))$, where $v_i$ is the predictive variance computed without $\mathbf{x}_i$.

$\longrightarrow$ Predictive variances are ignored in MSE-LOO.

The best criterion is problem and objective dependent.

# Analytical form

No approximation needed:
criteria can be computed in closed form.

| | | | |
|---|---|---|---|
| NLE | $\log\det K$ | $+$ | $Y^\top K^{-1} Y$ |
| NLP-LOO | $-\sum \log(K^{-1})_{ii}$ | $+$ | $\frac{(K^{-1}Y)_i^2}{(K^{-1})_{ii}}$ |
| MSE-LOO | | | $\sum \frac{(K^{-1}Y)_i^2}{(K^{-1})_{ii}^2}$ |

**Claim**

The success of Gaussian Processes for regression comes from this closed form solution.

# Analytical form

No approximation needed:
criteria can be computed in closed form.

| | | | |
|---|---|---|---|
| NLE | $\log\det K$ | $+$ | $Y^\top K^{-1} Y$ |
| NLP-LOO | $-\sum \log(K^{-1})_{ii}$ | $+$ | $\frac{(K^{-1}Y)_i^2}{(K^{-1})_{ii}}$ |
| MSE-LOO | | | $\sum \frac{(K^{-1}Y)_i^2}{(K^{-1})_{ii}^2}$ |

**Claim**

The success of Gaussian Processes for regression comes from this closed form solution.

# Toy experiment

## Prior mismatch

- Squared exponential covariance function, $K(\mathbf{x}_i, \mathbf{x}_j) = a \exp\left(-\theta ||\mathbf{x}_i - \mathbf{x}_j||^2\right) + \sigma^2 \delta_{ij}$.
- Target function = step function with Gaussian noise.

## Experimental setup

- The three hyperparameters $a$, $\sigma$ and $\theta$ are learned by minimizing the NLE.
- $a$ and $\sigma$ fixed. $\theta$ minimizing NLP-LOO / MSE-LOO is optimized in $[\theta_{NLE}/2 \ldots 2\ \theta_{NLE}]$.
- Ratio of the true MSE achieved by these 3 different criteria to the best MSE (for $\theta$ in the same interval).
- Experiments repeated 100 times.

# Toy experiment

## Prior mismatch

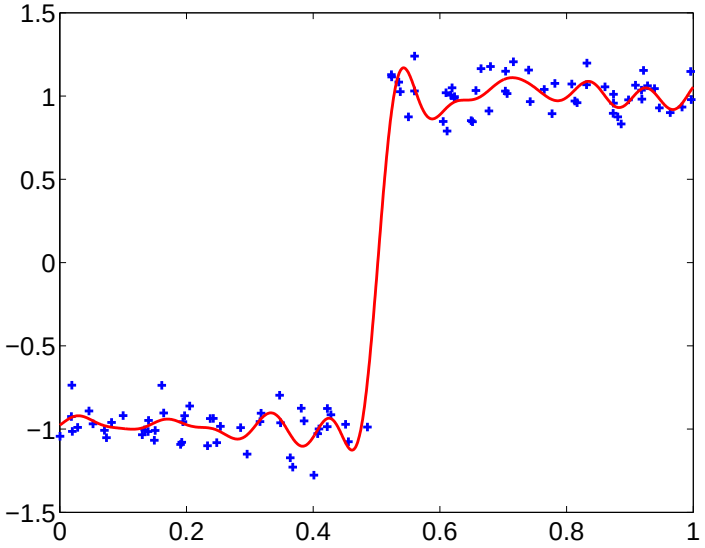
- Squared exponential covariance function, $K(\mathbf{x}_i, \mathbf{x}_j) = a \exp\left(-\theta ||\mathbf{x}_i - \mathbf{x}_j||^2\right) + \sigma^2 \delta_{ij}$.
- Target function = step function with Gaussian noise.

## Experimental setup

- The three hyperparameters $a$, $\sigma$ and $\theta$ are learned by minimizing the NLE.
- $a$ and $\sigma$ fixed. $\theta$ minimizing NLP-LOO / MSE-LOO is optimized in $[\theta_{NLE}/2 \ldots 2\ \theta_{NLE}]$.
- Ratio of the true MSE achieved by these 3 different criteria to the best MSE (for $\theta$ in the same interval).
- Experiments repeated 100 times.

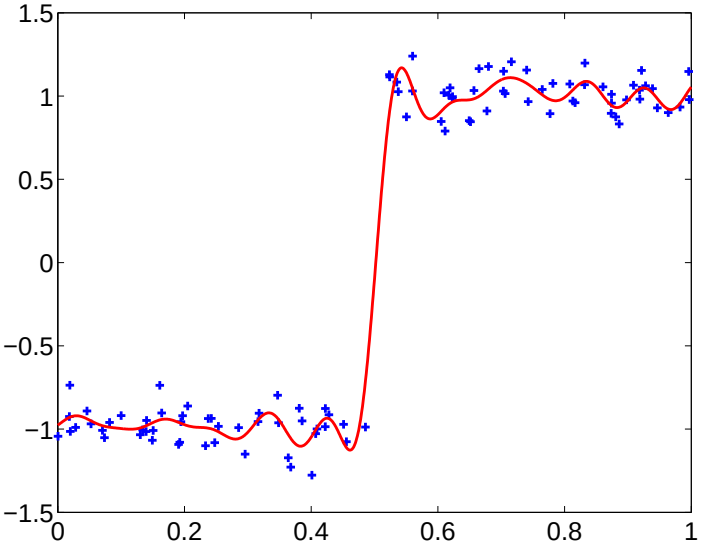
# Results

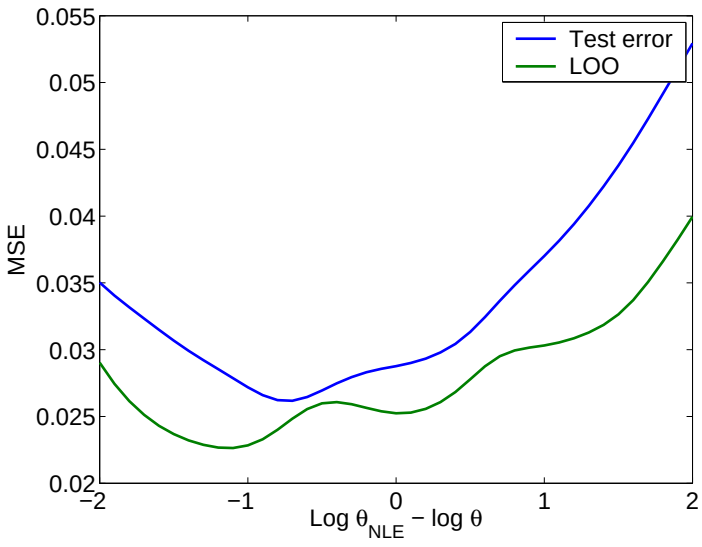
- 100 points
- Noise variance $= 10^{-2}$.
- Hyperparameters
  - $a = 0.7 \longrightarrow$ too small
  - $\sigma^2 = 0.014 \longrightarrow$ OK
  - $\theta = 253$ (larger is better)
- LOO generally less smooth

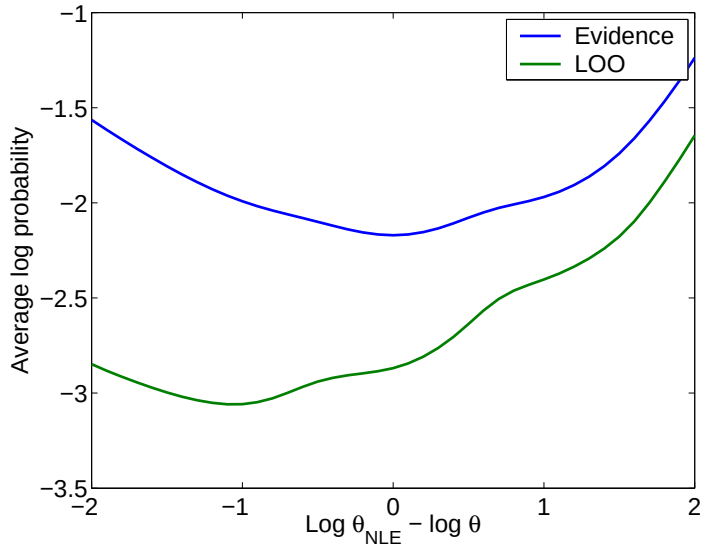


# Results

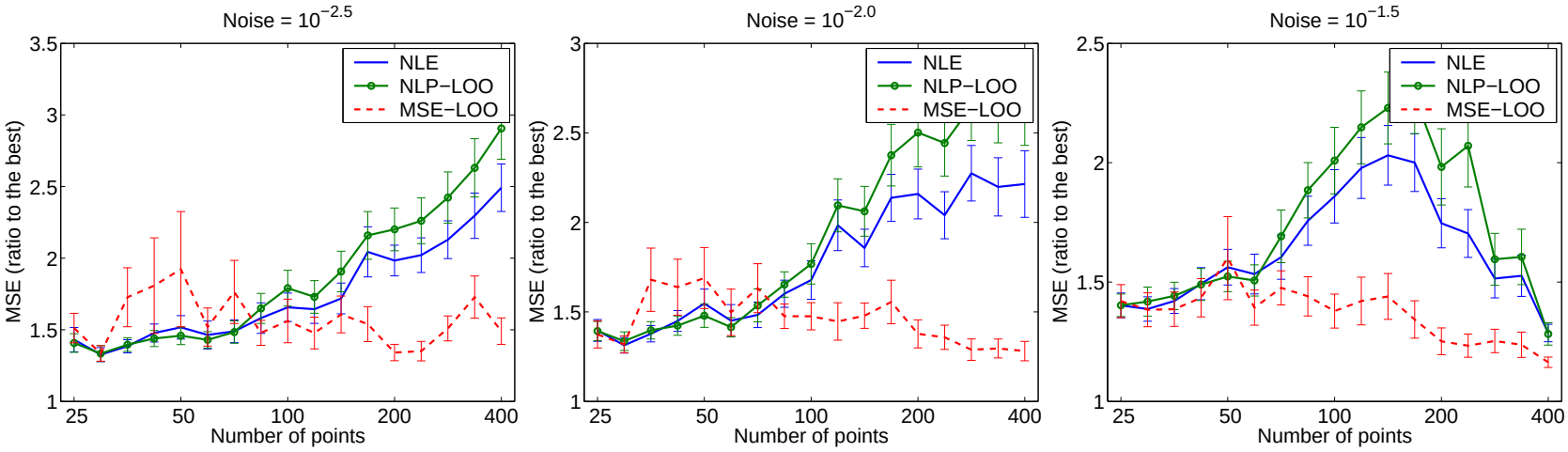
- 100 points
- Noise variance $= 10^{-2}$.
- Hyperparameters
  - $a = 0.7 \longrightarrow$ too small
  - $\sigma^2 = 0.014 \longrightarrow$ OK
  - $\theta = 253$ (larger is better)
- LOO generally less smooth

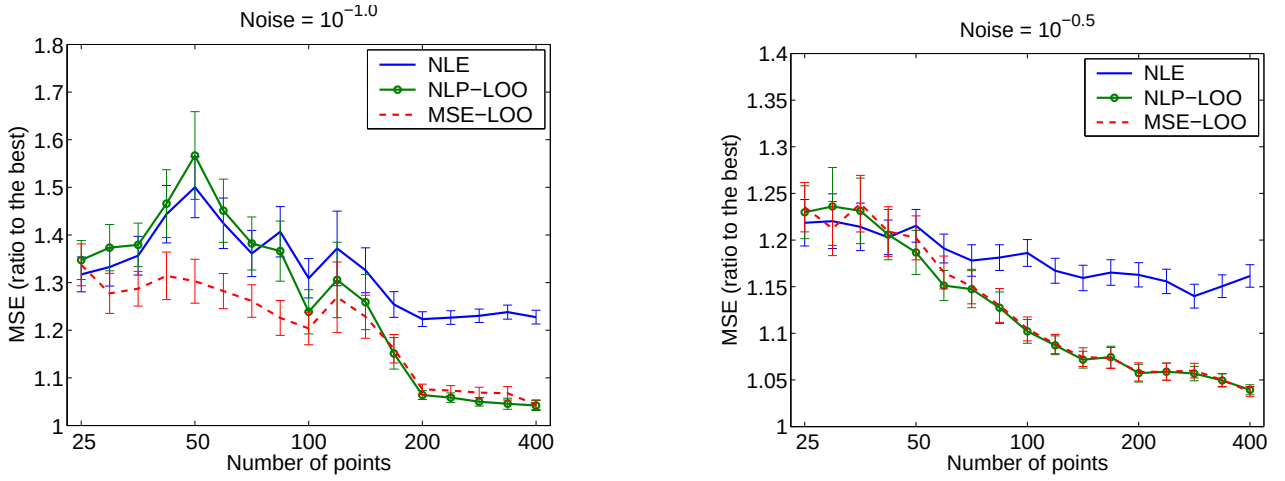


# Results


Noise = $10^{-2.5}$
Noise = $10^{-2.0}$
Noise = $10^{-1.5}$
NLE
NLP−LOO
MSE−LOO
MSE (ratio to the best)
Number of points


Noise = $10^{-1.0}$
Noise = $10^{-0.5}$
NLE
NLP−LOO
MSE−LOO
MSE (ratio to the best)
Number of points

# Summing up

## My conclusions

- If you really trust your prior, you should do Bayesian model selection, i.e. evidence maximization.
- If not, cross-validation techniques are a useful safeguard.

## Additional remarks

- In my opinion, this conclusion applies to Bayesian inference in general.
- Cross-validation errors seem more difficult to optimize (lots of local minimum). Mixed strategies could be useful.

# Summing up

## My conclusions

- If you really trust your prior, you should do Bayesian model selection, i.e. evidence maximization.
- If not, cross-validation techniques are a useful safeguard.

Additional remarks

- In my opinion, this conclusion applies to Bayesian inference in general.
- Cross-validation errors seem more difficult to optimize (lots of local minimum). Mixed strategies could be useful.

# Motivations

## Time complexity

For $n$ training points, complexity is:

training $O(n^3)$ [matrix inversion]

testing $O(n)$ for the mean and $O(n^2)$ for the variance

$\longrightarrow$ When $n$ is large, approximations are required.

It is important to know whether the main motivation is to improve the training or testing time.

For instance, if the concern is about training time, a sparse method for which finding the basis functions is relatively expensive might not be relevant.

$\longrightarrow$ Here the motivation is training time reduction.

# Motivations

## Time complexity

For $n$ training points, complexity is:

training $O(n^3)$ [matrix inversion]

testing $O(n)$ for the mean and $O(n^2)$ for the variance

$\longrightarrow$ When $n$ is large, approximations are required.

It is important to know whether the main motivation is to improve the training or testing time.

For instance, if the concern is about training time, a sparse method for which finding the basis functions is relatively expensive might not be relevant.

$\longrightarrow$ Here the motivation is training time reduction.

# Two important classes of methods

## Sparse approximations

- A lot of recent work on this topic.
- Usually, forward greedy selection of basis functions according to a given criterion.
- With $k$ basis functions, the complexity is $O(nk^2)$ for training and $O(k)$ for the mean prediction ($O(k^2)$ for the variance) $\longrightarrow$ same as in a finite model with $k$ variables.

## Conjugate gradient

- Solve the linear system only approximatively with conjugate gradient optimization.
- Very useful when matrix vector multiplication is fast.
- Less work in this direction (cf the *Skilling* method).
- Testing is still $O(n)$ and variance is expensive to compute.

# Two important classes of methods

## Sparse approximations

- A lot of recent work on this topic.
- Usually, forward greedy selection of basis functions according to a given criterion.
- With $k$ basis functions, the complexity is $O(nk^2)$ for training and $O(k)$ for the mean prediction ($O(k^2)$ for the variance) $\longrightarrow$ same as in a finite model with $k$ variables.

## Conjugate gradient

- Solve the linear system only approximatively with conjugate gradient optimization.
- Very useful when matrix vector multiplication is fast.
- Less work in this direction (cf the *Skilling* method).
- Testing is still $O(n)$ and variance is expensive to compute.

# Two important classes of methods

## Sparse approximations

- A lot of recent work on this topic.
- Usually, forward greedy selection of basis functions according to a given criterion.
- With $k$ basis functions, the complexity is $O(nk^2)$ for training and $O(k)$ for the mean prediction ($O(k^2)$ for the variance) $\longrightarrow$ same as in a finite model with $k$ variables.

## Conjugate gradient

- Solve the linear system only approximatively with conjugate gradient optimization.
- Very useful when matrix vector multiplication is fast.
- Less work in this direction (cf the *Skilling* method).
- Testing is still $O(n)$ and variance is expensive to compute.

# Sparse methods

**Claim**

Random selection of basis functions is almost optimal.

Maybe a bit more basis functions are needed but time is saved by not optimizing their location.

**Claim**

When $k \ll n$, all unknown (hyper)-parameters and basis locations can be optimized by minimizing the MSE.

This is because we are in an underfitting situation.

So when $n$ is really large, the situation is very simple: we are back to good old RBF networks. More interesting from a ML point of view is when $n/k$ is, let's say, of the order of 10.

# Sparse methods

**Claim**

Random selection of basis functions is almost optimal.

Maybe a bit more basis functions are needed but time is saved by not optimizing their location.

**Claim**

When $k \ll n$, all unknown (hyper)-parameters and basis locations can be optimized by minimizing the MSE.

This is because we are in an underfitting situation.

So when $n$ is really large, the situation is very simple: we are back to good old RBF networks. More interesting from a ML point of view is when $n/k$ is, let's say, of the order of 10.

# Sparse methods

**Claim**

Random selection of basis functions is almost optimal.

Maybe a bit more basis functions are needed but time is saved by not optimizing their location.

**Claim**

When $k \ll n$, all unknown (hyper)-parameters and basis locations can be optimized by minimizing the MSE.

This is because we are in an underfitting situation.

So when $n$ is really large, the situation is very simple: we are back to good old RBF networks. More interesting from a ML point of view is when $n/k$ is, let's say, of the order of 10.

# Conjugate gradient

If the matrix has no special structure and $k$ iterations of CG are performed, training time is $O(kn^2)$.

Approximations yielding a special structure (and fast matrix vector multplication which is the core of a CG iteration):

- KD-trees, Fast Multipole, $\mathcal{H}$-matrices
- Low rank matrices (found by incomplete Cholesky decomposition)
- Sparse matrix from compactly supported kernel or from thresholding

low dimension

large bandwidth

small bandwidth

# Conjugate gradient

If the matrix has no special structure and $k$ iterations of CG are performed, training time is $O(kn^2)$.

Approximations yielding a special structure (and fast matrix vector multplication which is the core of a CG iteration):

- KD-trees, Fast Multipole, $\mathcal{H}$-matrices — low dimension
- Low rank matrices (found by incomplete Cholesky decomposition) — large bandwidth
- Sparse matrix from compactly supported kernel or from thresholding — small bandwidth

# Conjugate gradient

If the matrix has no special structure and $k$ iterations of CG are performed, training time is $O(kn^2)$.

Approximations yielding a special structure (and fast matrix vector multplication which is the core of a CG iteration):

- KD-trees, Fast Multipole, $\mathcal{H}$-matrices — low dimension
- Low rank matrices (found by incomplete Cholesky decomposition) — large bandwidth
- Sparse matrix from compactly supported kernel or from thresholding — small bandwidth

# Conjugate gradient

If the matrix has no special structure and $k$ iterations of CG are performed, training time is $O(kn^2)$.

Approximations yielding a special structure (and fast matrix vector multplication which is the core of a CG iteration):

- KD-trees, Fast Multipole, $\mathcal{H}$-matrices — low dimension
- Low rank matrices (found by incomplete Cholesky decomposition) — large bandwidth
- Sparse matrix from compactly supported kernel or from thresholding — small bandwidth

# Remarks on model selection

Main focus = tractable approximation of the predictive mean.

## Open questions

How does the rest of the Bayesian machinery follow ?
What is the evidence and does it make sense to maximize it ?

Since the data is abundant, one can just keep a fraction of it as a validation set.

# Remarks on model selection

Main focus = tractable approximation of the predictive mean.

## Open questions

How does the rest of the Bayesian machinery follow ?
What is the evidence and does it make sense to maximize it ?

Since the data is abundant, one can just keep a fraction of it as a validation set.