Lecture 03: Kernel regression and time series analysis

Gustavo Camps-Valls

Image Processing Laboratory (IPL) – Universitat de Valncia. Spain.
gustavo.camps@uv.es, http://www.uv.es/gcamps

The organization of the course:

1. Fundamentals of kernel methods
2. Supervised and unsupervised kernel-based classification
3. Kernel methods for regression and time series analysis <<<
4. Nonlinear feature extraction with kernels

Definitions

Definitions

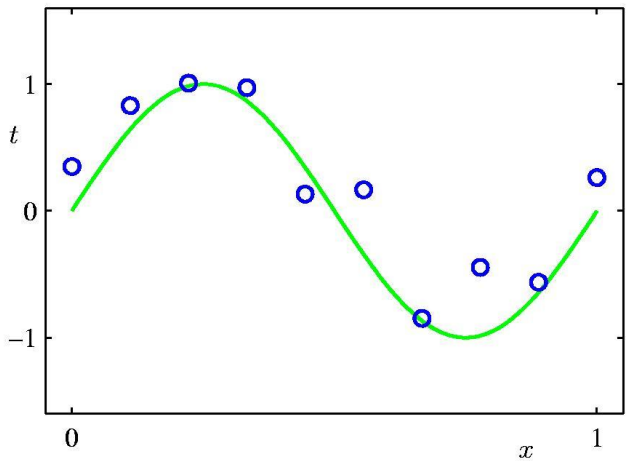
- Regression, curve fitting, function approximation
- Regression models involve the following variables:
  - The unknown parameters (weights) denoted as $\mathbf{w}$
  - The independent (input, features) variables, $\mathbf{x}$
  - The dependent (output, target) variable(s), $\mathbf{y}$
- A regression model f relates $\mathbf{y}$ with $\mathbf{x}$:

$$\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{w}) + \mathbf{e}$$

Definitions

Regression, curve fitting, function approximation

- Approximate m-dimensional continuous functions



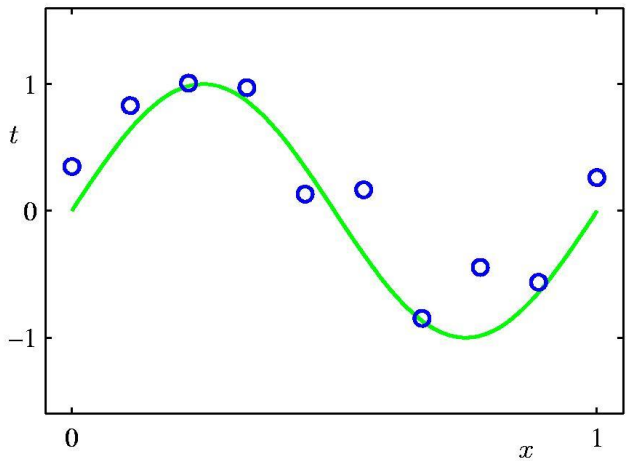
- To estimate a real function:

$$\begin{aligned} \mathbf{R}^n &\longrightarrow \mathbf{R}^m \\ \mathbf{x} &\longrightarrow \mathbf{y} = f(\mathbf{x}, \mathbf{w}) \end{aligned}$$

Definitions

Regression, curve fitting, function approximation

- Approximate m-dimensional continuous functions



- To estimate a real function:

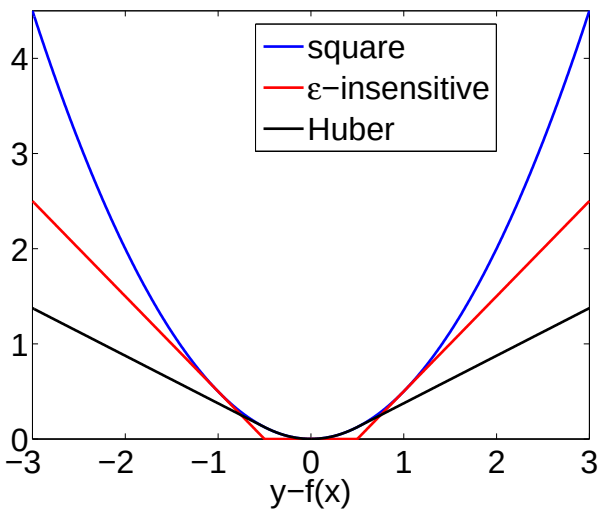
$$\begin{array}{rcl} \mathbf{R}^n & \longrightarrow & \mathbf{R}^m \\ \mathbf{x} & \longrightarrow & \mathbf{y} = f(\mathbf{x}, \mathbf{w}) \end{array}$$

- **How to choose the weights?**
  - √ Perform cross-validation to select parameters
  - √ Choose a quality (performance, objective) criterion to optimize
  - ? Minimize a cost function and regularize it

Risk, loss, cost, objective, energy

Risk, loss, cost, objective, energy

Loss	$\ell(\mathbf{y}, f(\mathbf{x}, \mathbf{w}))$
Squared loss	$(\mathbf{y} - f(\mathbf{x}, \mathbf{w}))^2$
Absolute loss	$\lvert\mathbf{y} - f(\mathbf{x}, \mathbf{w})\rvert$
ε-insensitive loss	$(\lvert\mathbf{y} - f(\mathbf{x}, \mathbf{w})\rvert - \varepsilon)_+$
Huber loss	$\begin{cases} \frac{1}{2}\mathbf{e}^2 & \lvert\mathbf{e}\rvert \leq \delta \\ \delta(\lvert\mathbf{e}\rvert - \frac{\delta}{2}) & \textit{otherwise} \end{cases}$
Log-cosh loss	$\ell(\mathbf{y}, f(\mathbf{x}, \mathbf{w})) = \log(\cosh(\mathbf{y} - f(\mathbf{x}, \mathbf{w})))$



Risk, loss, cost, objective, energy

Loss	$\ell(\mathbf{y}, f(\mathbf{x}, \mathbf{w}))$	**Noise model**, $p(\mathbf{e})$
Squared loss	$0.5(\mathbf{y} - f(\mathbf{x}, \mathbf{w}))^2$	$\frac{1}{\sqrt{2\pi}} \exp(-\mathbf{e}^2/2)$
Absolute loss, Laplacian	$\lvert\mathbf{y} - f(\mathbf{x}, \mathbf{w})\rvert$	$\frac{1}{2} \exp(-\lvert\mathbf{e}\rvert)$
ε-insensitive loss	$(\lvert\mathbf{y} - f(\mathbf{x}, \mathbf{w})\rvert - \varepsilon)_+$	$\frac{1}{\sqrt{2(1+\varepsilon)}} \exp(-\lvert\mathbf{e}\rvert_\varepsilon)$
Huber loss	$\begin{cases} \frac{1}{2}\mathbf{e}^2 & \lvert\mathbf{e}\rvert \leq \delta \\ \delta(\lvert\mathbf{e}\rvert - \frac{\delta}{2}) & \textit{otherwise} \end{cases}$	$\begin{cases} \exp(-\mathbf{e}^2/2) & \lvert\mathbf{e}\rvert \leq \delta \\ \exp(\delta/2 - \lvert\mathbf{e}\rvert) & \textit{otherwise} \end{cases}$

Maximum likelihood (ML): 'find the most likely function -model- that generated the data'

$$\max_{\mathbf{w}}\{p(\mathbf{x}, y|\mathbf{w})\} = \max_{\mathbf{w}}\left\{\prod_i p(\mathbf{x}_i, y_i|\mathbf{w})\right\} = \max_{\mathbf{w}}\left\{\prod_i p(y_i|\mathbf{x}_i, \mathbf{w})p(\mathbf{x}_i)\right\}$$

$$= \max_{\mathbf{w}}\left\{\prod_i p(y_i|\mathbf{x}_i, \mathbf{w})\right\} \approx \max_{\mathbf{w}}\left\{-\log(p(\mathbf{x}, y|\mathbf{w}))\right\}$$

Regularizer

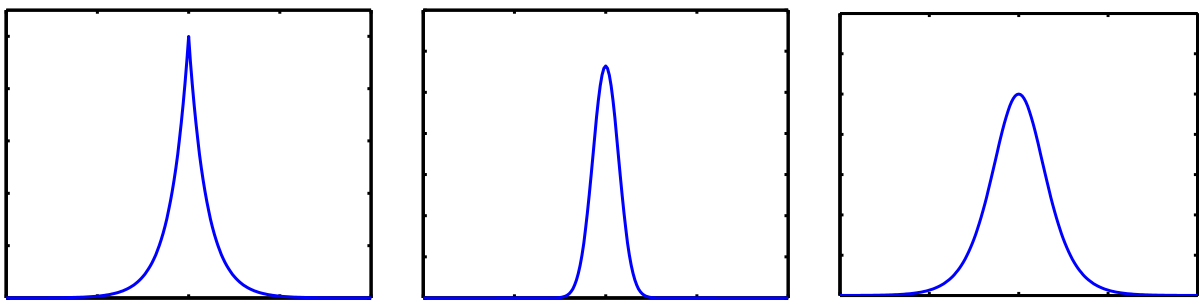
- Regularization is used to prevent overfitting and simplify the solution:

Regularizer

- Regularization is used to prevent overfitting and simplify the solution:
  - Implicitly or explicitly penalize models based on the number of effective parameters or directions

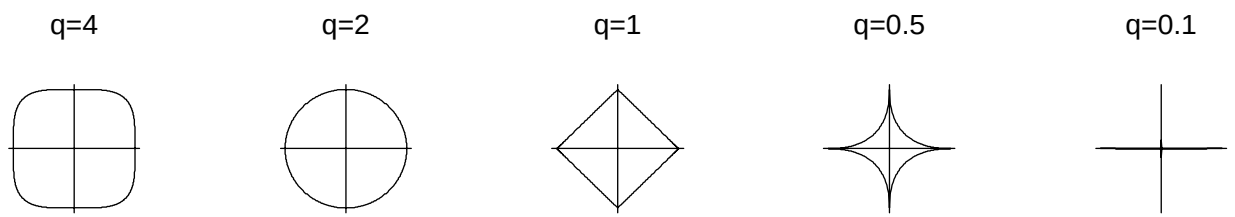
Regularizer

- Regularization is used to prevent overfitting and simplify the solution:
  - Implicitly or explicitly penalize models based on the number of effective parameters or directions
  - Bayesian methods use a *prior probability* that (usually) gives lower probability to more complex models



Regularizer

Model	Loss	Regularizer (entropy measure)
AIC/BIC	$(\mathbf{y} - f(\mathbf{x}, \mathbf{w}))^2$	$\|\mathbf{w}\|_0$
Ridge regression, KRR, GP	$(\mathbf{y} - f(\mathbf{x}, \mathbf{w}))^2$	$\|\mathbf{w}\|_2$
SVR	$\|\mathbf{y} - f(\mathbf{x}, \mathbf{w})\|_\varepsilon$	$\|\mathbf{w}\|_2$
Lasso	$(\mathbf{y} - f(\mathbf{x}, \mathbf{w}))^2$	$\|\mathbf{w}\|_1$
Basis pursuit denoising	$(\mathbf{y} - f(\mathbf{x}, \mathbf{w}))^2$	$\|\mathbf{w}\|_1$
Reg Least Absolute Deviations	$\|\mathbf{y} - f(\mathbf{x}, \mathbf{w})\|$	$\|\mathbf{w}\|_1$



$$\|\mathbf{w}\|_q = \sum_j |\mathbf{w}_i|^q$$

Regularized linear regression

Regularized least squares linear regression

- Add a ridge penalty to the loss function:

$$\min_{\mathbf{W}} \left\{ \|\mathbf{Y} - \mathbf{X}\mathbf{W}\|^2 + \lambda \|\mathbf{W}\|^2 \right\}$$

- The weights (one per feature) are obtained easily

$$\hat{\mathbf{W}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}$$

Homework

1. Demonstrate the normal equations for the regularized case (above)
2. Do a Matlab function that, given $\mathbf{X}$ and $\mathbf{Y}$ automatically:
   - returns the v-fold results
   - the optimal λ parameter

Kernel Regularized linear regression

Exercise: Kernelize the regularized least squares solution:

- Model: $\mathbf{Y} = \mathbf{XW}$
- Functional:

$$\mathbf{W}^* = \min_{\mathbf{W}} \left\{ \|\mathbf{Y} - \mathbf{XW}\|^2 + \|\mathbf{W}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \mathbf{X}^\dagger \mathbf{Y}$

Kernel Regularized linear regression

Exercise: Kernelize the regularized least squares solution:

- Model: $\mathbf{Y} = \mathbf{X}\mathbf{W}$
- Functional:

$$\mathbf{W}^* = \min_{\mathbf{W}} \left\{ \|\mathbf{Y} - \mathbf{X}\mathbf{W}\|^2 + \|\mathbf{W}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y} = \mathbf{X}^\dagger \mathbf{Y}$

Hint

- Model: $\mathbf{Y} = \mathbf{\Phi}\mathbf{W}$
- Functional:

$$\mathbf{W}^*_{\mathcal{H}} = \min_{\mathbf{W}_{\mathcal{H}}} \left\{ \|\mathbf{Y} - \mathbf{\Phi}\mathbf{W}\|^2 + \|\mathbf{W}\|^2 \right\}$$

Kernel Regularized linear regression

Exercise: Kernelize the regularized least squares solution:

- Model: $\mathbf{Y} = \mathbf{XW}$
- Functional:

$$\mathbf{W}^* = \min_{\mathbf{W}} \left\{ \|\mathbf{Y} - \mathbf{XW}\|^2 + \|\mathbf{W}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \mathbf{X}^\dagger \mathbf{Y}$

Hint

- Model: $\mathbf{Y} = \mathbf{\Phi W}$
- Functional:

$$\mathbf{W}^*_{\mathcal{H}} = \min_{\mathbf{W}_{\mathcal{H}}} \left\{ \|\mathbf{Y} - \mathbf{\Phi W}\|^2 + \|\mathbf{W}\|^2 \right\}$$

Hint2

Derive, use the representer theorem and apply simple matrix manipulation

Kernel Regularized linear regression

Regularized least squares linear classification

- Inputs: $\mathbf{X} \in \mathbb{R}^{n \times d}$
- Outputs: $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^\top$
- Model: $\mathbf{Y} = \mathbf{X}\mathbf{w}$
- Functional:

$$\mathbf{w}^* = \min_{\mathbf{w}} \left\{ \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{Y}$

Kernel Regularized linear regression

Regularized least squares linear classification

- Inputs: $\mathbf{X} \in \mathbb{R}^{n \times d}$
- Outputs: $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^\top$
- Model: $\mathbf{Y} = \mathbf{X}\mathbf{w}$
- Functional:

$$\mathbf{w}^* = \min_{\mathbf{w}} \left\{ \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{Y}$

Regularized kernel least squares classification

- Model: $\mathbf{Y} = \mathbf{\Phi}\mathbf{w}_{\mathcal{H}}$
- Functional:

$$\mathbf{w}^*_{\mathcal{H}} = \min_{\mathbf{w}_{\mathcal{H}}} \left\{ \|\mathbf{Y} - \mathbf{\Phi}\mathbf{w}_{\mathcal{H}}\|^2 + \lambda \|\mathbf{w}_{\mathcal{H}}\|^2 \right\}$$

- Dual weights: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{Y}$
- Primal weigths: $\mathbf{w}_{\mathcal{H}} = \mathbf{\Phi}^\top \boldsymbol{\alpha}$
- Decision function $\mathbf{Y} = \mathbf{\Phi}\mathbf{w}_{\mathcal{H}} = \mathbf{K}\boldsymbol{\alpha}$

Kernel Regularized linear regression

Exercise: Kernel ridge regression

- Solve the 'normal equations' in feature spaces
- Assume squared cost function
- Regularize the weights

Solution

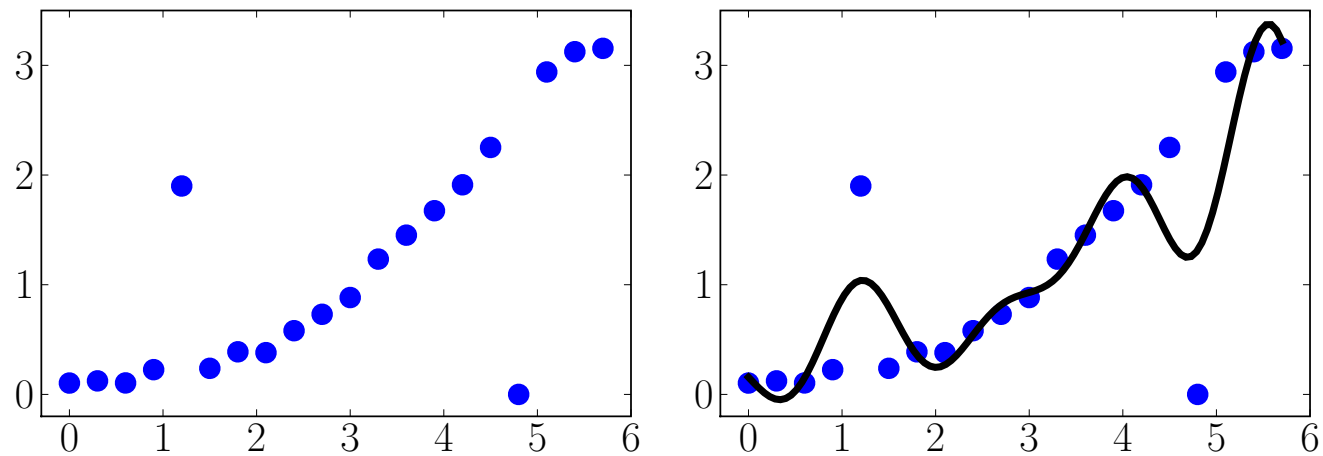
- $\boldsymbol{\alpha} = (\lambda \mathbf{I} + \mathbf{K})^{-1}\mathbf{y}$
- $\hat{\mathbf{y}} = \mathbf{K}\boldsymbol{\alpha}$

Matlab

```
>> sigma=3; gamma=1e-5
>> K = kernelmatrix('rbf',X,X,sigma);
>> alpha = inv(gamma*eye(length(Y)) + K)*Y;
>> Ypred = alpha*K;
>> assessment(Y,Ypred,'regress')
```

KRR problems and solutions

Like Least-Squared Regression, (Kernel) Ridge Regression is sensitive to outliers:



because the quadratic loss function penalized large residue.

KRR problems and solutions

Problems!

- One weight per example $\longrightarrow$ Risk of overfitting
- High computational cost for $n > 2000$, different (σ, λ) to try!

KRR problems and solutions

Problems!

- One weight per example $\rightarrow$ Risk of overfitting
- High computational cost for $n > 2000$, different (σ, λ) to try!

Solutions

- Do proper cross validation and control λ. Plot λ- RMSE_{test} !!!
- Standard code: `alpha = inv(gamma + K) * Y;`
- Cholesky decomposition is faster ($\sim$4-fold) but not sparse again:
  ```
  R = chol(K+gamma*eye(n));
  alpha = R\(R'\Y);
  ```
- Nyström method uses the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A} + \mathbf{V}\mathbf{V}^\top)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{D}^{-1}$$

KRR problems and solutions

Problems!

- One weight per example $\rightarrow$ Risk of overfitting
- High computational cost for $n > 2000$, different (σ, λ) to try!

Solutions

- Do proper cross validation and control λ. Plot λ- RMSE_{test} !!!
- Standard code: `alpha = inv(gamma + K) * Y;`
- Cholesky decomposition is faster ($\sim$4-fold) but not sparse again:
  ```
  R = chol(K+gamma*eye(n));
  alpha = R\(R'\Y);
  ```
- Nyström method uses the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A} + \mathbf{V}\mathbf{V}^\top)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{D}^{-1}$$

- There are tricks to make the KRR sparse

Reduced-rank (Sparse) KRR

A sparsification procedure [Cawley04]

- Find a reduced basis in $\mathcal{H}$
- Method for training a reduced rank method

Step 1: Form a basis in feature space

- Define a subset of the vectors, $\phi_S(\mathbf{x}_i) \equiv \{\phi(\mathbf{x}_i)\}_{i \in S}$
- Compute the reconstruction error for every sample

$$\delta_i = \frac{\|\phi(\mathbf{x}_i) - \phi_S(\mathbf{x}_i)\|^2}{\|\phi(\mathbf{x}_i)\|^2} = 1 - \frac{\mathbf{K}_{Si}^\top \mathbf{K}_{SS}^{-1} \mathbf{K}_{Si}}{\mathbf{K}_{ii}}$$

- The basis is built by minimizing the reconstruction error δ_i, i.e. maximize:

$$\mathcal{L}(S) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbf{K}_{Si}^\top \mathbf{K}_{SS}^{-1} \mathbf{K}_{Si}}{\mathbf{K}_{ii}}$$

- Greedy algorithm: start with $S = [0]$, maximize $\mathcal{L}(S)$, stop when $\mathbf{K}_{SS}$ singular

Reduced-rank (Sparse) KRR

A sparsification procedure [Cawley04]

- Find a reduced basis in $\mathcal{H}$
- Method for training a reduced rank method

Step 2: Training procedure

- Model: $\mathbf{Y} = \mathbf{\Phi}\mathbf{w}_S$
- The selected basis induces a reduced rank representer's theorem:

$$\mathbf{w}_S = \sum_{i \in S} \beta_i \phi(\mathbf{x}_i) = \mathbf{\Phi}_S^\top \boldsymbol{\beta}$$

- The KRR functional to minimize slightly changes

$$\mathbf{w}_S^* = \min_{\mathbf{w}_S} \left\{ \|\mathbf{Y} - \mathbf{\Phi}\mathbf{w}_S\|^2 + \lambda \|\mathbf{w}_S\|^2 \right\}$$

- So the dual weights:

$$\boldsymbol{\beta} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{Y}$$

$$\boldsymbol{\beta} = (\mathbf{K}_{:S}^\top \mathbf{K}_{S:} + \lambda \mathbf{I}_{SS})^{-1} \mathbf{K}_{:S}^\top \mathbf{Y}$$

- Primal weights: $\mathbf{w}_S = \mathbf{\Phi}_S^\top \boldsymbol{\beta}$
- Decision function $\mathbf{Y} = \mathbf{\Phi}\mathbf{w}_S = \mathbf{K}_{:,s}\boldsymbol{\beta}$

SVR

- SVR also works in a supervised way
- We have input-output data pairs $\{(\mathbf{x}_i, y_i),\ i = 1, ..., n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.
- First, map the data and then do a linear regression in the feature space:

$$\phi : \mathbb{R}^d \rightarrow \mathbb{R}^H,\ d \leq H$$

- The linear model in $\mathcal{H}$:

$$\hat{y}_i = f(\mathbf{x}_i, \mathbf{w}) = \phi^T(\mathbf{x}_i)\mathbf{w} + b$$

  where $\mathbf{w}$ is the weight vector and b is the bias term
- We assume and additive noise model:

$$y_i = \hat{y}_i + e_i$$

  where e_i are the committed errors by the model

SVR

- SVR also works in a supervised way
- We have input-output data pairs $\{(\mathbf{x}_i, y_i),\ i = 1, ..., n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.
- First, map the data and then do a linear regression in the feature space:

$$\phi : \mathbb{R}^d \rightarrow \mathbb{R}^H,\ d \leq H$$

- The linear model in $\mathcal{H}$:

$$\hat{y}_i = f(\mathbf{x}_i, \mathbf{w}) = \phi^T(\mathbf{x}_i)\mathbf{w} + b$$

  where $\mathbf{w}$ is the weight vector and b is the bias term
- We assume and additive noise model:

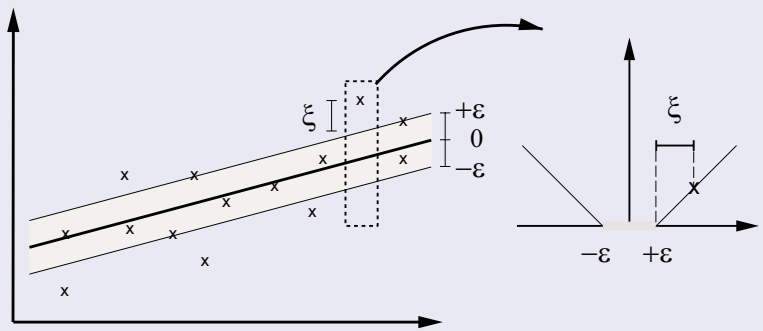
$$y_i = \hat{y}_i + e_i$$

  where e_i are the committed errors by the model
- **What's new versus KRR?**

SVR

The SVR formulation

- The ε-insensitive SVR is the SVM implementation for regression and function approximation



- The (regularized) primal functional to be minimized is:

$$\min_{\mathbf{w},\xi,b}\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i(\xi_i + \xi_i^*)\}$$

with respect to $\mathbf{w}$, ξ_i, ξ_i^* and b, subject to:

$$\begin{aligned} y_i - \phi^T(\mathbf{x}_i)\mathbf{w} - b &\leq \varepsilon + \xi_i & \forall i = 1,\ldots,n \\ \phi^T(\mathbf{x}_i)\mathbf{w} + b - y_i &\leq \varepsilon + \xi_i^* & \forall i = 1,\ldots,n \\ \xi_i, \xi_i^* &\geq 0 & \forall i = 1,\ldots,n \end{aligned}$$

SVR

- Introduce restrictions in the primal through Lagrange multipliers:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i(\xi_i + \xi_i^*) - \sum_i \alpha_i(\varepsilon - y_i + \phi^T(\mathbf{x}_i)\mathbf{w} + b)$$

$$- \sum_i \alpha_i^*(\varepsilon + y_i - \phi^T(\mathbf{x}_i)\mathbf{w} - b) - \sum_i(\mu_i\xi_i + \mu_i^*\xi_i^*)$$

- Derive and set to zero:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\phi(\mathbf{x}_i) = 0$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \mu_i, = 0,\ \ i = 1, ..., L$$

$$\frac{\partial L_p}{\partial \xi_i^*} = C - \alpha_i^* - \mu_i^* = 0,\ \ i = 1, ..., L$$

- Lagrange multipliers must be positive: $\alpha_i, \alpha_i^*, \mu_i, \mu_i^* \geq 0$

SVR

- The dual (Wolfe's) problem becomes:

$$L_d = \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^{n}(\alpha_i + \alpha_i^*)$$
$$-\frac{1}{2}\sum_{i,j}^{n}(\alpha_i - \alpha_i^*)(\alpha_i + \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}_j)$$

s.t.

$$C \geq \alpha_i^{(*)} \geq 0,$$

SVR

- The dual (Wolfe's) problem becomes:

$$L_d = \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^{n}(\alpha_i + \alpha_i^*)$$
$$-\frac{1}{2}\sum_{i,j}^{n}(\alpha_i - \alpha_i^*)(\alpha_i + \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}_j)$$

  s.t.

$$C \geq \alpha_i^{(*)} \geq 0,$$

- Predictions:

$$\hat{y}_i = f(\mathbf{x}_i, \mathbf{w}) = \phi^T(\mathbf{x}_i)\mathbf{w} + b = \sum_i (\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}) + b$$

SVR in Matlab

- **The SVR problem is a QP problem with linear constraints**
- The dual problem in matrix form:

$$(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{y} - \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) - \frac{1}{2} (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{K} (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$$

s.t.

$$C \geq \alpha_i^{(*)} \geq 0,$$

where $\boldsymbol{\alpha}^{(*)} = \left[\alpha_1^{(*)}, \cdots, \alpha_n^{(*)}\right]^T$ and $\mathbf{y} = [y_1, \cdots, y_n]^T$.

- Use this:

```
>> help quadprog
QUADPROG Quadratic programming.
X=QUADPROG(H,f,A,b) attempts to solve the quadratic programming problem:
min 0.5*x'*H*x + f'*x subject to:  A*x <= b
x
```

- Identify terms: $\mathrm{x} := (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T$ and $H := K$ and passing it to MATLAB:

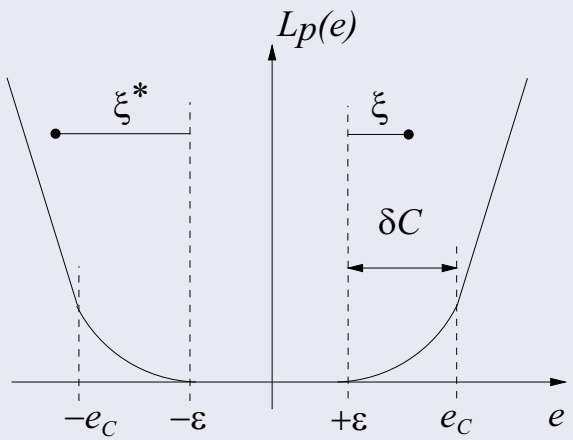
```
alphas = quadprog(H,-Y'+e*f,[],[],f1,0,zeros(size(Y')),C*f,[],OPTIONS);
```

- The bias term b is obtained from averaging some SVs.
- This is cool: sparsity in $\boldsymbol{\alpha} - \boldsymbol{\alpha}^*$ means interpretability!

A Robust SVR formulation

- Change the ε-insensitive SVR cost function to accomodate three kinds of noise sources (insensitively, quadratic, linear):

$$L(e_i) = \begin{cases} 0, & |\, e_i \,| \leq \varepsilon \\ \frac{1}{2\delta}(|\, e_i \,| - \varepsilon)^2, & \varepsilon \leq |\, e_i \,| \leq e_C \\ C(|\, e_i \,| - \varepsilon) - \frac{1}{2}\delta C^2, & |\, e_i \,| \geq e_C \end{cases}$$



- "Robust Support Vector Regression for Biophysical Parameter Estimation from Remotely Sensed Images" Gustavo Camps-Valls, L. Bruzzone, Jose L. Rojo-Ivarez, Farid Melgani IEEE Geoscience and Remote Sensing Letters, July 2006. Volume: 3, Issue: 3, pp. 339- 343

H-SVR

```
>> R=[R -R;-R R];
>> Y=Y(:);
>> f1=[ones(size(Y')) -ones(size(Y'))];
>> Y=[Y;-Y];
>> H=(R+delta*eye(size(R,1)));
>> unos=ones(size(Y'));
>> OPTIONS = optimset('LargeScale','on','diffmaxchange',1e-8,...
'Diagnostics','off','Display','off');
>>
alpha=quadprog(H,-Y'+e*unos,[],[],f1,0,zeros(size(Y')),C*unos,[],OPTIONS);
```

Data collection and experimental setup

In this work, we use two different datasets:

1. **MERIS dataset**.
   - SIMULATED DATA = NO UNCERTAINTY
   - 1000 samples for cross-validating, 4000 for testing.
2. **SeaBAM dataset**.
   - REAL DATA = UNCERTAINTY
   - 460 samples for cross-validating, 460 for testing.

Numerical Results

Table: Mean error (ME), root mean-squared error (RMSE), mean absolute error (ABSE), and correlation coefficient (r) of models in the test set.

	ME	RMSE	ABSE	r
MERIS database				
ε-SVR	-2.36e-4	0.015	0.061	0.998
Squared loss SVR	-9.96e-4	0.031	0.018	0.998
ε-Huber-SVR	-3.26e-6	0.011	0.004	0.999
SeaBAM database				
ε-SVR	-0.070	0.139	0.105	0.971
Squared loss SVR	-0.034	0.140	0.107	0.971
ε-Huber-SVR	-0.020	0.137	0.104	0.972

- ε-Huber-SVR is more accurate (RMSE, ABSE) and unbiased (ME) than the rest of the models
- For the MERIS data, appreciable numerical and statistical differences.
- For the SeaBAM dataset, the proposed method showed an improved numerical performance but no statistical differences.

H-SVR

Robustness to reduced datasets

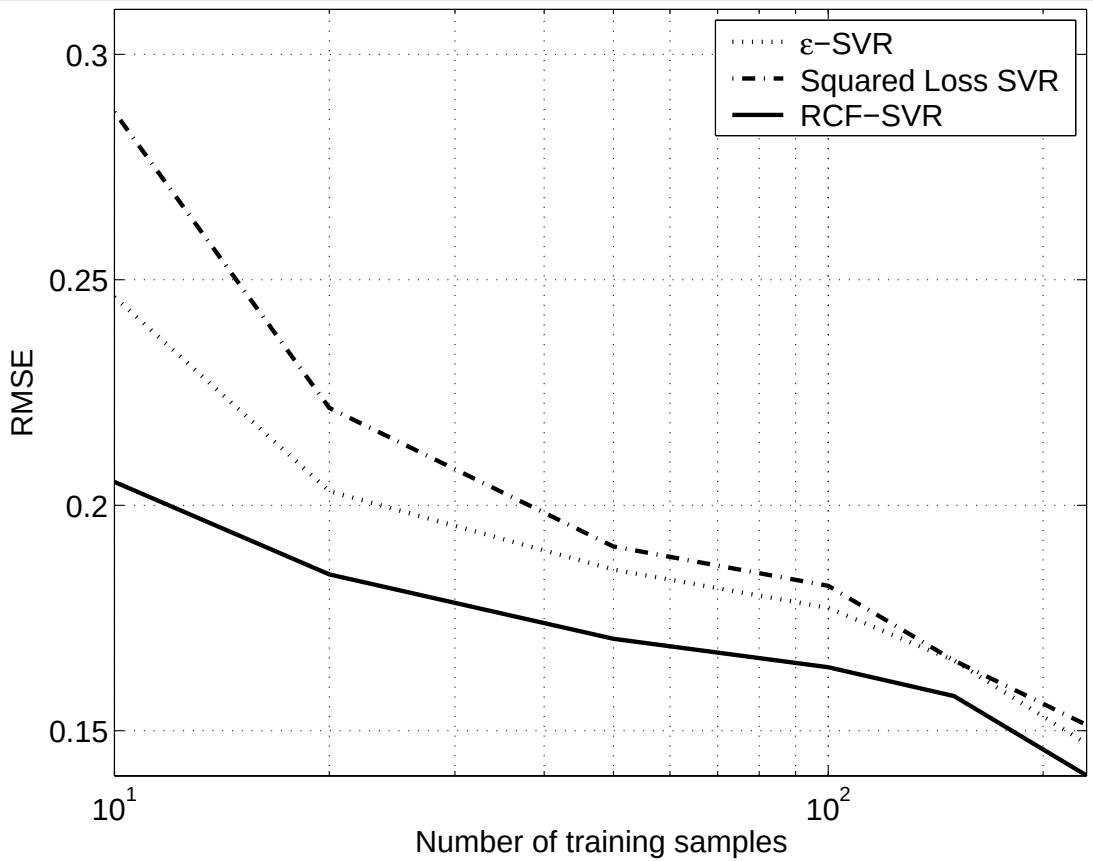


Figure: Evolution of the RMSE in the test SeaBAM set as a function of the number of training samples.

PD-SVR

1. Efficiency of oligonucleotides in RNA sequences
2. Drug concentration prediction
3. Image coding

"Predictions are hard, particularly those concerning the future..."

Andreas S. Weigend, 1990.

Motivation

- Independent and identically distributed (i.i.d.) signals (or random variables)
- Time series, speech and images are not iid signals
- How to:
  - Define SVM methods for TSA?
  - Kernelize linear structures with the kernel trick?
  - How to define static and dynamic (online) kernel methods?

Motivation

- Independent and identically distributed (i.i.d.) signals (or random variables)
- Time series, speech and images are not iid signals
- How to:
  - Define SVM methods for TSA?
  - Kernelize linear structures with the kernel trick?
  - How to define static and dynamic (online) kernel methods?

Concepts

- Autoregressive and moving average (ARMA) processes
- FIR and IIR filters
- Adaptive filters
- Auto-correlation, memory depth and temporal resolution

Motivation

- Independent and identically distributed (i.i.d.) signals (or random variables)
- Time series, speech and images are not iid signals
- How to:
  - Define SVM methods for TSA?
  - Kernelize linear structures with the kernel trick?
  - How to define static and dynamic (online) kernel methods?

Concepts

- Autoregressive and moving average (ARMA) processes
- FIR and IIR filters
- Adaptive filters
- Auto-correlation, memory depth and temporal resolution

Outline

- SVM-ARMA
- Kernel ARMA

Independent and identically distributed (i.i.d.) random variables

- **i.i.d random variables:** if each random variable has the same probability distribution as the others and all are mutually independent.
- **aka, Exchangeable random variables:** sequence such that future samples behave like earlier samples, i.e.: any order is equally likely

Independent and identically distributed (i.i.d.) random variables

- **i.i.d random variables:** if each random variable has the same probability distribution as the others and all are mutually independent.
- **aka, Exchangeable random variables:** sequence such that future samples behave like earlier samples, i.e.: any order is equally likely
- **Why?** This assumption typically simplifies many formulations

Independent and identically distributed (i.i.d.) random variables

- **i.i.d random variables:** if each random variable has the same probability distribution as the others and all are mutually independent.
- **aka, Exchangeable random variables:** sequence such that future samples behave like earlier samples, i.e.: any order is equally likely
- **Why?** This assumption typically simplifies many formulations
- **Examples of iid signals:**
  - roulette wheel
  - dice rolls
  - coin flips

Independent and identically distributed (i.i.d.) random variables

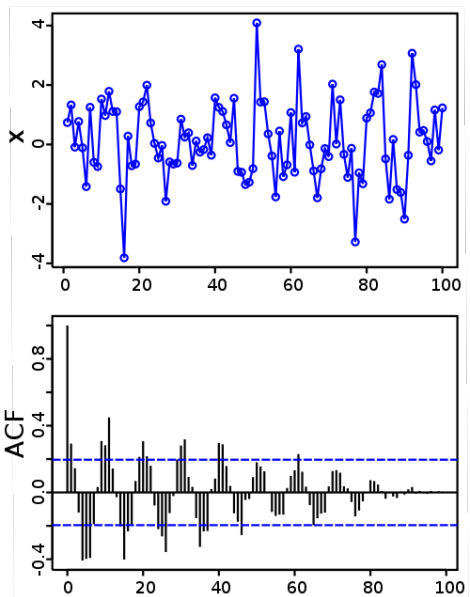
- **i.i.d random variables:** if each random variable has the same probability distribution as the others and all are mutually independent.
- **aka, Exchangeable random variables:** sequence such that future samples behave like earlier samples, i.e.: any order is equally likely
- **Why?** This assumption typically simplifies many formulations
- **Examples of iid signals:**
  - roulette wheel
  - dice rolls
  - coin flips
- **Examples of non-iid signals:**
  - exchange rates
  - speech
  - image/video sequences

(Auto)correlation function

- **Auto-correlation function ('correlogram'):** Given a stationary process $\{x_t\}$, the ACF is

$$\rho_x(h) = corr(x_{t+h}, x_t) = \sum_k x(k)x(k+h)$$

```
>> x=sin(1:100)+randn(1,100); stem(xcorr(x))
```



- Signal processing: Identify pulsar events, tempo, beats, pitch
- Image processing: extent and period of pixel relations in space

Auto-regressive and Moving-average (ARMA) process

- **ARMA modeling.** Given the input-output time series $\{x_n\}$ and $\{y_n\}$

$$y_n = \underbrace{\sum_{i=1}^{P} a_i y_{n-i}}_{AR} + \underbrace{\sum_{j=1}^{Q} b_j x_{n-j+1}}_{MA} + e_n, n = 1, \ldots, N$$

Auto-regressive and Moving-average (ARMA) process

- **ARMA modeling.** Given the input-output time series $\{x_n\}$ and $\{y_n\}$

$$y_n = \underbrace{\sum_{i=1}^{P} a_i y_{n-i}}_{AR} + \underbrace{\sum_{j=1}^{Q} b_j x_{n-j+1}}_{MA} + e_n, n = 1, \ldots, N$$

- The error terms e_n are generally assumed to be iid sampled from a normal distribution with zero mean, $\mathcal{N}(0, \sigma_n^2)$

Auto-regressive and Moving-average (ARMA) process

- **ARMA modeling.** Given the input-output time series $\{x_n\}$ and $\{y_n\}$

$$y_n = \underbrace{\sum_{i=1}^{P} a_i y_{n-i}}_{AR} + \underbrace{\sum_{j=1}^{Q} b_j x_{n-j+1}}_{MA} + e_n, n = 1, \ldots, N$$

- The error terms e_n are generally assumed to be iid sampled from a normal distribution with zero mean, $\mathcal{N}(0, \sigma_n^2)$
- After choosing P and Q, just obtain a_i and b_j by least squares regression:

$$\min_{\mathbf{a},\mathbf{b}} \{\|\mathbf{y} - \mathbf{Ya} - \mathbf{Xb}\|^2\} = \min_{\mathbf{w}} \{\|\mathbf{y} - \mathbf{Zw}\|^2\}$$

where $\mathbf{y} \in \mathbb{R}^{n\times 1}$, $\mathbf{Y} \in \mathbb{R}^{n\times P}$, $\mathbf{a} \in \mathbb{R}^{P\times 1}$, $\mathbf{X} \in \mathbb{R}^{n\times Q}$ and $\mathbf{b} \in \mathbb{R}^{Q\times 1}$.

Auto-regressive and Moving-average (ARMA) process

- **ARMA modeling.** Given the input-output time series $\{x_n\}$ and $\{y_n\}$

$$y_n = \underbrace{\sum_{i=1}^{P} a_i y_{n-i}}_{AR} + \underbrace{\sum_{j=1}^{Q} b_j x_{n-j+1}}_{MA} + e_n, n = 1, \ldots, N$$

- The error terms e_n are generally assumed to be iid sampled from a normal distribution with zero mean, $\mathcal{N}(0, \sigma_n^2)$
- After choosing P and Q, just obtain a_i and b_j by least squares regression:

$$\min_{\mathbf{a},\mathbf{b}} \{\|\mathbf{y} - \mathbf{Ya} - \mathbf{Xb}\|^2\} = \min_{\mathbf{w}} \{\|\mathbf{y} - \mathbf{Zw}\|^2\}$$

  where $\mathbf{y} \in \mathbb{R}^{n\times 1}$, $\mathbf{Y} \in \mathbb{R}^{n\times P}$, $\mathbf{a} \in \mathbb{R}^{P\times 1}$, $\mathbf{X} \in \mathbb{R}^{n\times Q}$ and $\mathbf{b} \in \mathbb{R}^{Q\times 1}$.
- Overfitting naively controlled by choosing low P and Q values

Auto-regressive and Moving-average (ARMA) process

- **ARMA modeling.** Given the input-output time series $\{x_n\}$ and $\{y_n\}$

$$y_n = \underbrace{\sum_{i=1}^{P} a_i y_{n-i}}_{AR} + \underbrace{\sum_{j=1}^{Q} b_j x_{n-j+1}}_{MA} + e_n, n = 1, \ldots, N$$

- The error terms e_n are generally assumed to be iid sampled from a normal distribution with zero mean, $\mathcal{N}(0, \sigma_n^2)$
- After choosing P and Q, just obtain a_i and b_j by least squares regression:

$$\min_{\mathbf{a},\mathbf{b}}\{\|\mathbf{y} - \mathbf{Ya} - \mathbf{Xb}\|^2\} = \min_{\mathbf{w}}\{\|\mathbf{y} - \mathbf{Zw}\|^2\}$$

  where $\mathbf{y} \in \mathbb{R}^{n\times 1}$, $\mathbf{Y} \in \mathbb{R}^{n\times P}$, $\mathbf{a} \in \mathbb{R}^{P\times 1}$, $\mathbf{X} \in \mathbb{R}^{n\times Q}$ and $\mathbf{b} \in \mathbb{R}^{Q\times 1}$.
- Overfitting naively controlled by choosing low P and Q values
- M-estimates: regularization and time-varying cost functions ...

Auto-regressive and Moving-average (ARMA) process

- **ARMA modeling.** Given the input-output time series $\{x_n\}$ and $\{y_n\}$

$$y_n = \underbrace{\sum_{i=1}^{P} a_i y_{n-i}}_{AR} + \underbrace{\sum_{j=1}^{Q} b_j x_{n-j+1}}_{MA} + e_n, n = 1, \ldots, N$$

- The error terms e_n are generally assumed to be iid sampled from a normal distribution with zero mean, $\mathcal{N}(0, \sigma_n^2)$
- After choosing P and Q, just obtain a_i and b_j by least squares regression:

$$\min_{\mathbf{a},\mathbf{b}}\{\|\mathbf{y} - \mathbf{Ya} - \mathbf{Xb}\|^2\} = \min_{\mathbf{w}}\{\|\mathbf{y} - \mathbf{Zw}\|^2\}$$

  where $\mathbf{y} \in \mathbb{R}^{n\times 1}$, $\mathbf{Y} \in \mathbb{R}^{n\times P}$, $\mathbf{a} \in \mathbb{R}^{P\times 1}$, $\mathbf{X} \in \mathbb{R}^{n\times Q}$ and $\mathbf{b} \in \mathbb{R}^{Q\times 1}$.
- Overfitting naively controlled by choosing low P and Q values
- M-estimates: regularization and time-varying cost functions ...
- Matlab's sysid toolbox: `ar.m`, `arx.m`, `armax.m`, `pem.m`, etc.

(Linear) SVM-ARMA formulation: the Vapnik's cost

- Standard SVM for regression uses Vapnik's ε-insensitive loss ....

$$L_\varepsilon(e_n) = \begin{cases} |e_n| - \varepsilon, & \text{if } |e_n| \geq \varepsilon, \\ 0, & \text{if } |e_n| < \varepsilon \end{cases}$$

- ... and regularize model weights with the ℓ_2 norm:

$$L_P(a_i, b_j, e_n) = \frac{1}{2}\left(\sum_{i=1}^{P} a_i^2 + \sum_{j=1}^{Q} b_j^2\right) + C\sum_{n=k_o}^{N} L_\varepsilon(e_n)$$

- Primal problem:

$$L_P\left(a_i, b_j, \xi_n, \xi_n^*\right) = \frac{1}{2}\left(\sum_{i=1}^{P} a_i^2 + \sum_{j=1}^{Q} b_j^2\right) + C\sum_{n=k_o}^{N} \left(\xi_n + \xi_n^*\right)$$

subject to

$$\begin{aligned} y_n - \sum_{i=1}^{P} a_i y_{n-i} - \sum_{j=1}^{Q} b_j x_{n-j+1} &\leq \varepsilon + \xi_n \\ -y_n + \sum_{i=1}^{P} a_i y_{n-i} + \sum_{j=1}^{Q} b_j x_{n-j+1} &\leq \varepsilon + \xi_n^* \end{aligned}$$

(Linear) SVM-ARMA formulation: the Vapnik's cost

- Do:

$$\frac{\partial L_{PD}}{\partial a_i} = 0; \quad \frac{\partial L_{PD}}{\partial b_j} = 0; \quad \frac{\partial L_{PD}}{\partial \xi_n^{(*)}} = 0$$

- This gives:

$$0 \leq \alpha_n^{(*)} \leq C \tag{1}$$

$$a_i = \sum_{n=k_o}^{N} (\alpha_n - \alpha_n^*)\, y_{n-i} \tag{2}$$

$$b_j = \sum_{n=k_o}^{N} (\alpha_n - \alpha_n^*)\, x_{n-j+1} \tag{3}$$

- ... and also the input and output autocorrelation matrices emerge:

$$R_y^P(m,k) = \sum_{i=1}^{P} y_{m-i} y_{k-i}$$

$$R_x^Q(m,k) = \sum_{j=1}^{Q} x_{m-j+1} x_{k-j+1}$$

(Linear) SVM-ARMA formulation: the Vapnik's cost

- The dual problem becomes

$$L_D = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \left[\mathbf{R_x}^Q + \mathbf{R_y}^P\right](\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \\ + (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{y} - \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*)$$

- The QP problem, $\mathbf{z^T H z + b^T z}$, becomes

$$\begin{aligned} \mathbf{z} &= \left[\ \boldsymbol{\alpha}^T,\quad \boldsymbol{\alpha}^{*T}\ \right]^T \\ \mathbf{H} &= -\frac{1}{2}\begin{bmatrix} \mathbf{R_x}^Q + \mathbf{R_y}^P, & -\mathbf{R_x}^Q - \mathbf{R_y}^P \\ -\mathbf{R_x}^Q - \mathbf{R_y}^P, & \mathbf{R_x}^Q + \mathbf{R_y}^P \end{bmatrix} \\ \mathbf{b} &= \left[\ \mathbf{y}^T - \varepsilon,\quad -\mathbf{y}^T - \varepsilon\ \right]^T \end{aligned}$$

(Linear) SVM-ARMA formulation: the Vapnik's cost

- The dual problem becomes

$$L_D = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \left[\mathbf{R_x}^Q + \mathbf{R_y}^P\right](\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \\ + (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{y} - \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*)$$

- The QP problem, $\mathbf{z^T H z + b^T z}$, becomes

$$\begin{aligned} \mathbf{z} &= \left[\ \boldsymbol{\alpha}^T,\quad \boldsymbol{\alpha}^{*T}\ \right]^T \\ \mathbf{H} &= -\frac{1}{2}\begin{bmatrix} \mathbf{R_x}^Q + \mathbf{R_y}^P, & -\mathbf{R_x}^Q - \mathbf{R_y}^P \\ -\mathbf{R_x}^Q - \mathbf{R_y}^P, & \mathbf{R_x}^Q + \mathbf{R_y}^P \end{bmatrix} \\ \mathbf{b} &= \left[\ \mathbf{y}^T - \varepsilon,\quad -\mathbf{y}^T - \varepsilon\ \right]^T \end{aligned}$$

- **Clearly H is not invertible!**

(Linear) SVM-ARMA formulation: the Vapnik's cost

- The dual problem becomes

$$L_D = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \left[\mathbf{R_x}^Q + \mathbf{R_y}^P\right](\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \\ + (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{y} - \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*)$$

- The QP problem, $\mathbf{z^T H z + b^T z}$, becomes

$$\begin{aligned} \mathbf{z} &= \left[\ \boldsymbol{\alpha}^T,\quad \boldsymbol{\alpha}^{*T}\ \right]^T \\ \mathbf{H} &= -\frac{1}{2}\begin{bmatrix} \mathbf{R_x}^Q + \mathbf{R_y}^P, & -\mathbf{R_x}^Q - \mathbf{R_y}^P \\ -\mathbf{R_x}^Q - \mathbf{R_y}^P, & \mathbf{R_x}^Q + \mathbf{R_y}^P \end{bmatrix} \\ \mathbf{b} &= \left[\ \mathbf{y}^T - \varepsilon,\quad -\mathbf{y}^T - \varepsilon\ \right]^T \end{aligned}$$

- **Clearly H is not invertible!**
- **Regularization solves it: $\mathbf{H}' = \mathbf{H} + \gamma\mathbf{I}$**

(Linear) SVM-ARMA formulation: the Vapnik's cost

- So what we want to actually solve is:

$$L_D^{SVM} = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \left[\mathbf{R_x}^Q + \mathbf{R_y}^P\right](\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \\ + (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{y} - \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) - \\ -\frac{\gamma}{2}\left(\boldsymbol{\alpha}^T \mathbf{I} \boldsymbol{\alpha} + \boldsymbol{\alpha}^{*T} \mathbf{I} \boldsymbol{\alpha}^*\right)$$

(s.t. $0 \leq \alpha^{(*)} \leq C$).

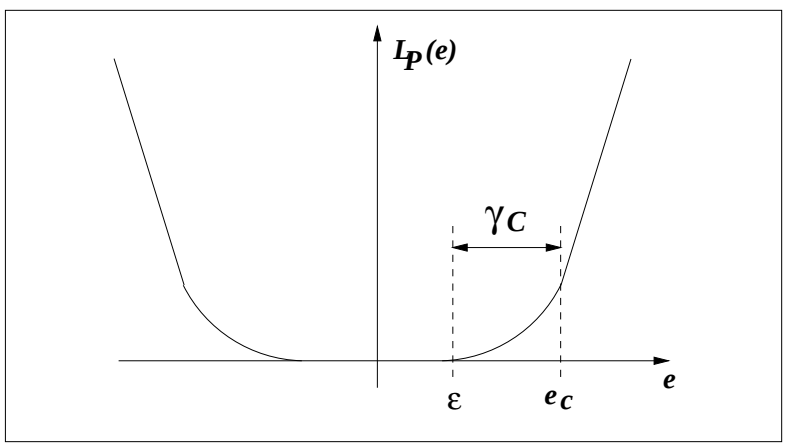
(Linear) SVM-ARMA formulation: the Vapnik's cost

- So what we want to actually solve is:

$$L_D^{SVM} = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \left[\mathbf{R_x}^Q + \mathbf{R_y}^P\right](\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \\ + (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{y} - \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) - \\ -\frac{\gamma}{2}\left(\boldsymbol{\alpha}^T \mathbf{I}\boldsymbol{\alpha} + \boldsymbol{\alpha}^{*T}\mathbf{I}\boldsymbol{\alpha}^*\right)$$

  (s.t. $0 \leq \alpha^{(*)} \leq C$).

- **This corresponds to changing the loss function!**



(Linear) SVM-ARMA formulation: the Vapnik's cost

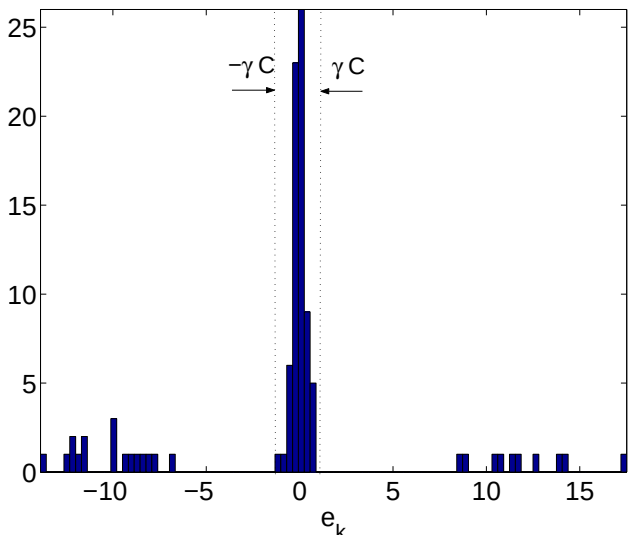
Example:

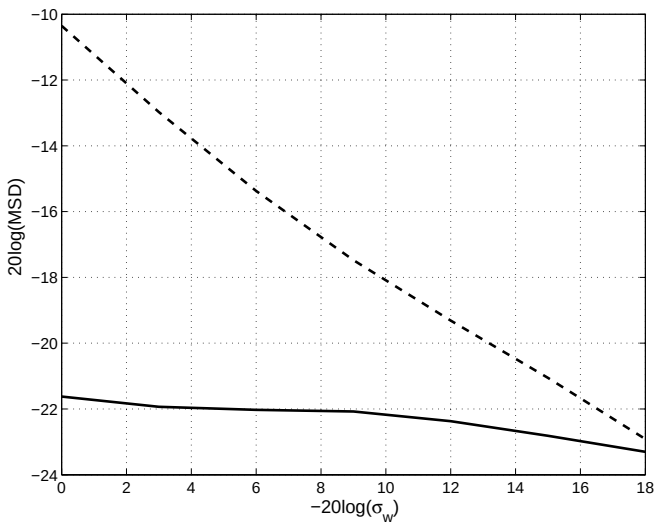
$$y_n = 0.03y_{n-1} - 0.01y_{n-2} + 3x_n - 0.5x_{n-1} + 0.2x_{n-2}$$

with

$$\{x_n\} \sim N(0,1), \{e_n\} \sim N(0,0.1), \{o_n\} = \{y_n\} + \{e_n\}$$

and impulsive noise $\{j_n\}$: 30% of samples with $\pm 10 + U(0,1)$





Kernelization of ARMA 1.

Kernelization of ARMA 1.

- The ARMA process can be vectorized, where $S = max(P, Q)$ and pad with zeroes the shortest series:

$$\mathbf{y}_n = [y_{n-1}, y_{n-2}, \ldots, y_{n-S}]^\top, \qquad \mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-S+1}]^\top$$

Kernelization of ARMA 1.

- The ARMA process can be vectorized, where $S = max(P, Q)$ and pad with zeroes the shortest series:

$$\mathbf{y}_n = [y_{n-1}, y_{n-2}, \ldots, y_{n-S}]^\top, \qquad \mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-S+1}]^\top$$

- The standard way is to use the SVR for regression ...
  1. Build/stack/encapsule the data: $\mathbf{z}_n = \left[\mathbf{y}_n^T, \mathbf{x}_n^T\right]^\top$
  2. Map them to $\mathcal{H}$ with $\phi(\mathbf{z}_n) : \mathbb{R}^{2S} \rightarrow \mathbb{R}^B$.
  3. Build a linear regression there: $y_n = \mathbf{d}^T \phi(\mathbf{z}_n) + e_n$

Kernelization of ARMA 1.

- The ARMA process can be vectorized, where $S = max(P, Q)$ and pad with zeroes the shortest series:

$$\mathbf{y}_n = [y_{n-1}, y_{n-2}, \ldots, y_{n-S}]^\top, \qquad \mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-S+1}]^\top$$

- The standard way is to use the SVR for regression ...
  1. Build/stack/encapsule the data: $\mathbf{z}_n = \left[\mathbf{y}_n^T, \mathbf{x}_n^T\right]^\top$
  2. Map them to $\mathcal{H}$ with $\phi(\mathbf{z}_n) : \mathbb{R}^{2S} \rightarrow \mathbb{R}^B$.
  3. Build a linear regression there: $y_n = \mathbf{d}^T \phi(\mathbf{z}_n) + e_n$
- The primal problem (using the ε-Huber loss) is then:

$$L_P^{SVM}\left(d_j, \xi_n^{(*)}\right) = \frac{1}{2}\sum_{j=1}^{B} d_j^2 + \frac{1}{2\gamma}\sum_{n\in I_1}\left(\xi_n^2 + \xi_n^{*2}\right)$$

$$+ C\sum_{n\in I_2}(\xi_n + \xi_n^*) - \sum_{n\in I_2}\frac{\gamma C^2}{2}$$

Kernelization of ARMA 1.

- The ARMA process can be vectorized, where $S = max(P, Q)$ and pad with zeroes the shortest series:

$$\mathbf{y}_n = [y_{n-1}, y_{n-2}, \ldots, y_{n-S}]^\top, \qquad \mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-S+1}]^\top$$

- The standard way is to use the SVR for regression ...
  1. Build/stack/encapsule the data: $\mathbf{z}_n = \left[\mathbf{y}_n^T, \mathbf{x}_n^T\right]^\top$
  2. Map them to $\mathcal{H}$ with $\phi(\mathbf{z}_n) : \mathbb{R}^{2S} \rightarrow \mathbb{R}^B$.
  3. Build a linear regression there: $y_n = \mathbf{d}^T \phi(\mathbf{z}_n) + e_n$
- The primal problem (using the ε-Huber loss) is then:

$$L_P^{SVM}\left(d_j, \xi_n^{(*)}\right) = \frac{1}{2}\sum_{j=1}^{B} d_j^2 + \frac{1}{2\gamma}\sum_{n\in I_1}\left(\xi_n^2 + \xi_n^{*2}\right)$$

$$+ C\sum_{n\in I_2}(\xi_n + \xi_n^*) - \sum_{n\in I_2}\frac{\gamma C^2}{2}$$

- The solution is the classical one

$$\hat{y}_n = \sum_{r=S}^{N}(\alpha_r - \alpha_r^*)K(\mathbf{z}_r, \mathbf{z}_n)$$

Kernelization of ARMA 2.

- Define ARMA in $\mathcal{H}$:

$$y_n = \mathbf{a}^T \phi(\mathbf{y}_n) + \mathbf{b}^T \phi(\mathbf{x}_n) + e_n$$

  with $\mathbf{a} = [a_1, \ldots, a_H]^\top$ and $\mathbf{b} = [b_1, \ldots, b_H]^\top$

- The primal problem minimizes:

$$L_P^{SVM}\left(a_i, b_i, \xi_n^{(*)}\right) = \frac{1}{2}\sum_{i=1}^{H}(a_i^2 + b_i^2) + \frac{1}{2\gamma}\sum_{n\in I_1}\left(\xi_n^2 + \xi_n^{*2}\right) +$$

$$+ C\sum_{n\in I_2}(\xi_n + \xi_n^*) - \sum_{n\in I_2}\frac{\gamma C^2}{2}$$

- The prediction model is now:

$$\hat{y}_n \quad = \sum_{r=S}^{N}(\alpha_r - \alpha_r^*)\left[K(\mathbf{y}_r, \mathbf{y}_n) + K(\mathbf{x}_r, \mathbf{x}_n)\right]$$

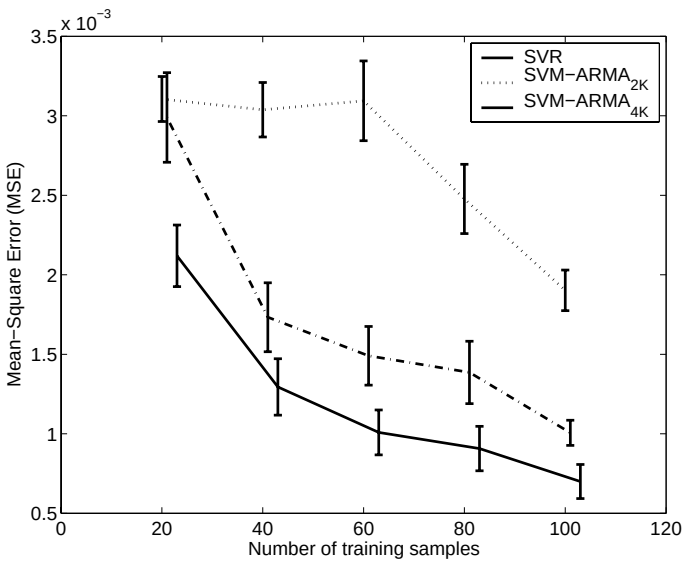
Example

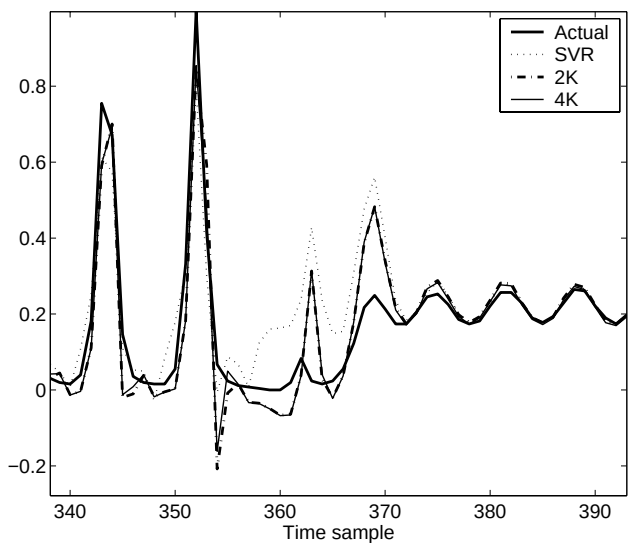
Example: `neon laser time series` (dataset A in the Santa Fe competition):

- 1-step ahead time series prediction problem:

$$y_n = f([y_{n-1}, y_{n-2}, ..., y_{n-k}], \mathbf{w})$$

- A complex transition from periodic to chaotic
- Noise-free, stationary, low dimensionality





Conclusions

- Given definition of the most useful kernel regression methods
- Other regression methods are available
- Analyzed how to derive the equations
- Multioutput SVR is not solved yet
- Kernel Bayesian approaches: RVM and GP
- Adaptive kernel learning is becoming very popular
- Everything relies on the proper definition of K

References

- J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.
- Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, NY, 1995.
- Vladimir Vapnik. *Statistical Learning Theory*. Wiley, NY, 1998.
- Ralf Herbrich. Learning Kernel Classifiers. MIT Press, Cambridge, MA, 2002.
- J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific Pub. Co., Singapore, 2002 `http://www.esat.kuleuven.be/sista/lssvmlab/`
- G. Camps-Valls, J. L. Rojo and M. Martinez, *Kernel Methods in Bioengineering, Signal and Image Processing*, Idea Inc., 2007.
- Conferences: NIPS, ICML, ECML, COLT, ICANN, ESANN, MLSP
- Webs: `videolectures.net`, `http://www.kernel-machines.org`