REGRESSION II

Machine Learning for Autonomous Robots

Manuel Meder
Robotics Group, University of Bremen

December 06, 2022 – Bremen, Deutschland

University of Bremen

German Research Center for Artificial Intelligence

1. Parameters
2. Support Vector Regression (SVR)
3. Gaussian Process Regression
4. Algorithm Class Perspective

Parameters

Types of parameters

Parameters

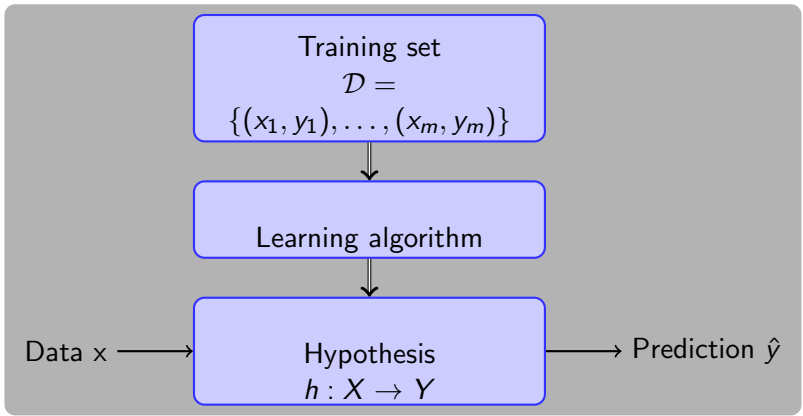
Parameters are the values the **algorithm will learn** using the given data. Paramters are therefore the result of the algorithm when being executed.

Hyper-Parameters

Are also known as meta-parameters. Those are values which the **expert (human) specifies** for a given algorithm in order to **tune or tweak the algorithms behaviour**.
Hyper-parameters are the set screws we use to fit our model best to our data. Finding the correct values of hyper-parameters is not easy and can partially be done by other algorithms.

Support Vector Regression

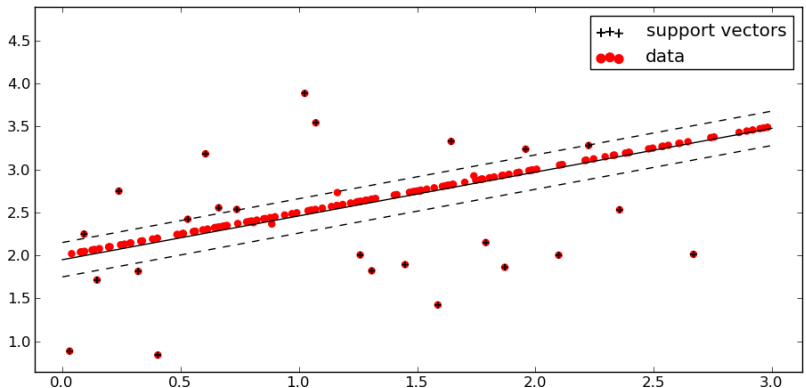
Recap: Supervised Learning



- If Y is a discrete domain: classification
- If Y is a continuous domain: regression

Idea: SV for Regression

- Learns a hyperplane h in kernel space that predicts most training data **approximately** correct
- Choose hyperplane such that a maximum number of training data points are within an ϵ-tube around the hyperplane



SVR: Properties

- We want to have a smooth function therefore we minimize $||w||_2^2$
- ϵ specifies the maximum allowed error in the prediction of an example that is not penalized
- Complexity C controls the trade-off between the flatness of h (small $||w||_2^2$) and the amount of deviations (ξ) larger than ϵ that are tolerable
- All points that lie within the ϵ-tube around the hypothesis do not affect the hypothesis (i.e. they could be moved arbitrarily within the tube without changing the learned hypothesis)

SVR: Formal definition

Optimization goal:

$$\min_w \left(\frac{1}{2} ||w||_2^2 \right) = \min_w \left(\frac{1}{2} w^T w \right)$$

- We want to penalize the deviation from the ϵ-tube by modeling those with ξ.
- As the problem is symmetricall we introduce ξ and ξ^* for deviations above and below the tube.
- In order to have a trade-off between a smooth function (complexity) and the penalty of deviations we use the complexity factor C.

Optimization goal:

$$\min_{w,\xi,\xi^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)^q$$

SVR: Formal definition

Optimization goal:

$$\min_{w,\xi,\xi^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)^q$$

We have some constraints which we have to take care of.

$$\begin{cases} -\epsilon - \xi_i^* \leq y_i - \langle w, x_i \rangle \\ y_i - \langle w, x_i \rangle \leq \epsilon + \xi_i \\ \xi_i \geq 0 \\ \xi_i^* \geq 0 \end{cases}$$

Those inequality constraints can be solved by rewriting them to ≥ 0 and adding them into the optimization problem using Lagrange multipliers.

SVR: Formal definition

For $q = 1$ we get the following optimization goal:

$$\min_{w,\xi,\xi^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} \epsilon + \xi_i^* + y_i - w^T x_i \geq 0 \\ \epsilon + \xi_i - y_i + w^T x_i \geq 0 \\ \xi_i \geq 0 \\ \xi_i^* \geq 0 \end{cases}$$

SVR: Formal definition

Hence we get:

$$\min_{w,\xi,\xi^*} \mathcal{L}(w,\xi,\xi^*,\alpha,\alpha*,\beta,\beta*) = \min_{w,\xi,\xi^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$
$$- \sum_{i=1}^{n} \alpha_i^* (\epsilon + \xi_i^* + y_i - w^T x_i) - \sum_{i=1}^{n} \alpha_i (\epsilon + \xi_i - y_i + w^T x_i)$$
$$- \sum_{i=1}^{n} \beta_i^* \xi_i^* - \sum_{i=1}^{n} \beta_i \xi_i$$

and maximize with respect to $\alpha, \alpha*, \beta, \beta*$.

To solve this and get rid of w, ξ, ξ^* we calculate the partial derivatives of the Lagrangian.

SVR: Formal definition

We get the following derivatives:

$$\nabla_w \mathcal{L} = w + \sum_{i=1}^{n} \alpha_i x_i - \sum_{i=1}^{n} \alpha_i^* x_i = 0$$

$$\Rightarrow w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^*} = C - \alpha_i^* - \beta_i^* = 0 \qquad \forall i$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \qquad \forall i$$

We can put back these equations into the original Lagrangian.

SVR: Formal definition

We get the following optimization goal:

$$\min_{w,\xi,\xi^*} \mathcal{L}(w,\xi,\xi^*,\alpha,\alpha*,\beta,\beta*) = \min_{w,\xi,\xi^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$
$$- \sum_{i=1}^{n} \alpha_i^* (\epsilon + \xi_i^* + y_i - w^T x_i) - \sum_{i=1}^{n} \alpha_i (\epsilon + \xi_i - y_i + w^T x_i)$$
$$- \sum_{i=1}^{n} \beta_i^* \xi_i^* - \sum_{i=1}^{n} \beta_i \xi_i$$

Since $C - \alpha_i - \beta_i = 0$ those terms collapse. Analogous for ξ_i^*

SVR: Formal definition

Also plugging in the term for w gives us:

$$\begin{aligned}\min_{w,\xi,\xi^*} \mathcal{L}(\dots) = \min_{w,\xi,\xi^*} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_j^T x_i \\ - \sum_{i=1}^{n} \alpha_i^* (\epsilon + y_i - \sum_{j=1}^{n} (\alpha_j - \alpha_j^*) x_j^T x_i) \\ - \sum_{i=1}^{n} \alpha_i (\epsilon - y_i + \sum_{j=1}^{n} (\alpha_j - \alpha_j^*) x_j^T x_i)\end{aligned}$$

We can now factor out ϵ and y_i.

SVR: Formal definition

Also plugging in the term for w gives us:

$$\begin{aligned}\min_{w,\xi,\xi^*} \mathcal{L}(\dots) = \min_{w,\xi,\xi^*} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x_j^T x_i \\ -\sum_{i=1}^{n}\alpha_i^*(-\sum_{j=1}^{n}(\alpha_j - \alpha_j^*)x_j^T x_i) \\ -\sum_{i=1}^{n}\alpha_i(\sum_{j=1}^{n}(\alpha_j - \alpha_j^*)x_j^T x_i) \\ +\sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \sum_{i=1}^{n}\epsilon(\alpha_i + \alpha_i^*)\end{aligned}$$

The double summed terms can be combined.

SVR: Formal definition

$$\min_{w,\xi,\xi^*} \mathcal{L}(\dots) = \min_{w,\xi,\xi^*} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_j^T x_i$$

$$-1 \times \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_j^T x_i)$$

$$+ \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \sum_{i=1}^{n} \epsilon(\alpha_i + \alpha_i^*)$$

And the final optimization term is:

$$\min_{w,\xi,\xi^*} -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_j^T x_i$$

$$+ \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \sum_{i=1}^{n} \epsilon(\alpha_i + \alpha_i^*)$$

SVR: Formal definition

From the initial problem (primal problem) we have reached now a term (equivalent dual problem for $q = 1$) that is free of w, ξ and ξ^* and only needs to be maximized with respect to the Lagrangian multipliers α and α^*. Note: β and β^* also dropped out.

$$\max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x_j^T x_i + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \sum_{i=1}^{n}\epsilon(\alpha_i + \alpha_i^*)$$

$$\text{subject to } \begin{cases} \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}$$

SVR: Formal definition

From the partial derivates we directly gain a formulation which constructs us our parameters w.
And therefore a formula to predict the value for new input values.

$$w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i \qquad f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i^T x$$

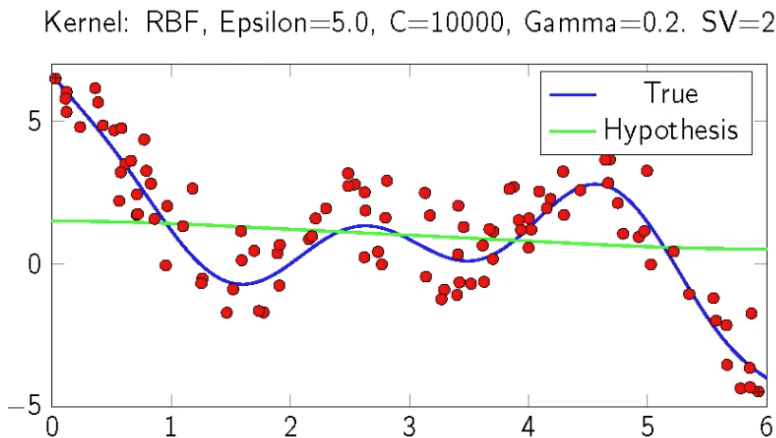
Each data point where one of the α's is not zero will contribute to the model (w). All others satisfy the condition with the ξ's and therefore are located within the ϵ-tube.

Notice: For prediction only the inner product of input values is needed.

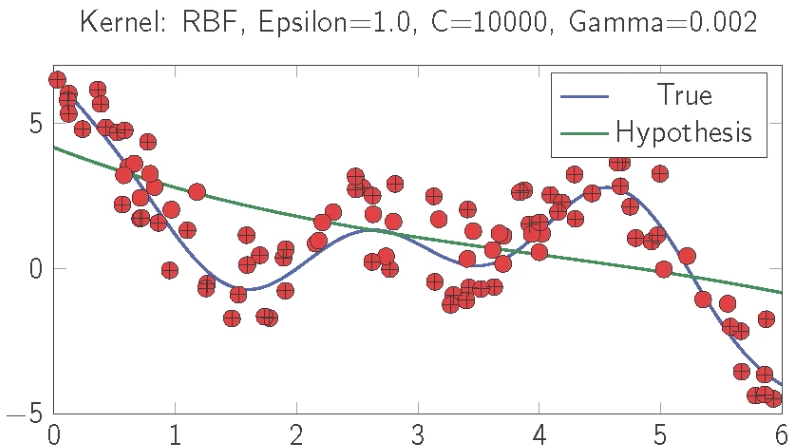
SVR: Kernels

- Kernel trick: replace inner product with a symmetric positive (semi-)definite function
- Chosen kernel determines the class of functions that can be learned
- Linear kernel $k(x_i, x_j) = \langle x_i, x_j \rangle \Rightarrow$ linear hypothesis
- Popular choice for kernels in SVR: radial basis function (RBF)
  $k(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||_2^2\right)$
- The hypothesis learned with the RBF kernel resembles those of non-parametric methods
- Parameter γ determines the "locality", i.e. how flat the learned hypothesis is

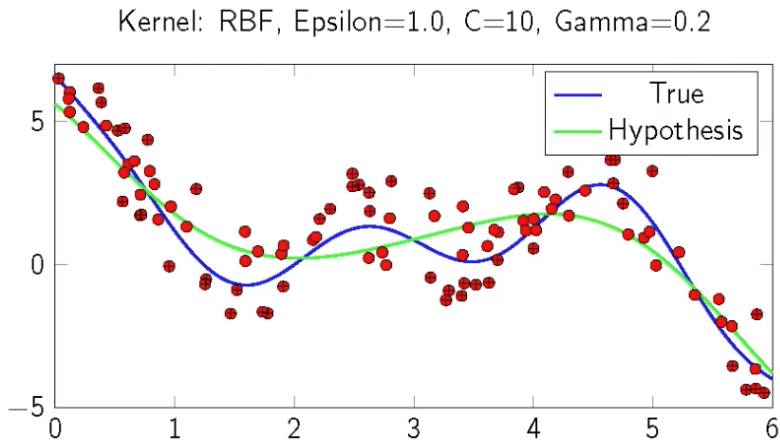
Example: Epsilon



Example: Gamma



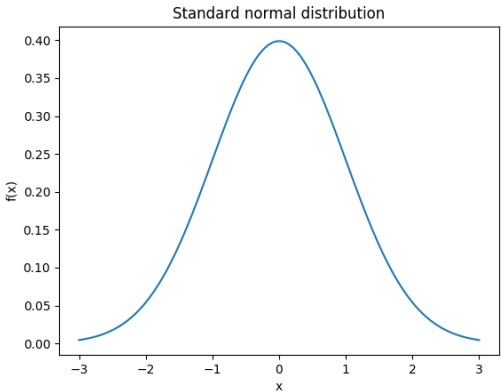
Example: Complexity



Gaussian Process Regression

Gaussian distribution

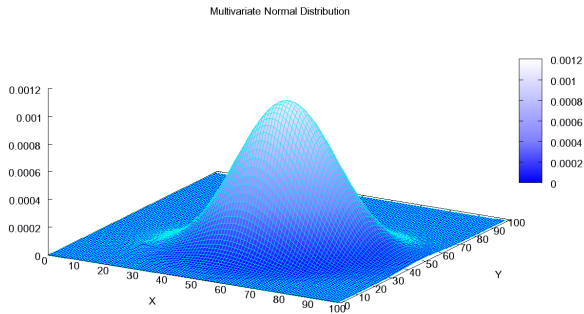
In this case $\mu = 0$ and $\sigma^2 = 1$ (standard normal distribution).



with $x \sim \mathcal{N}\left(\underbrace{0}_{\mu}, \underbrace{1}_{\sigma^2}\right)$ $\qquad f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{(x-\mu)^2}{2\sigma^2}$

Multivariate gaussian distribution

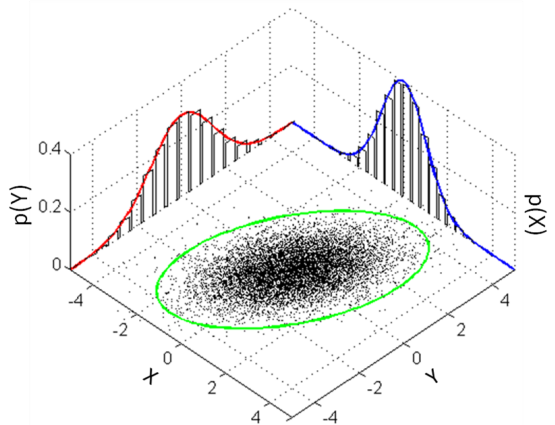
In this case bi-variate gaussian:



https://upload.wikimedia.org/wikipedia/commons/5/57/Multivariate_Gaussian.png

with $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$ $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (covariance matrix)

Marginal distribution of Gaussians



https://en.wikipedia.org/wiki/Marginal_distribution#/media/File:MultivariateNormal.png

with $p(x) = \int_y p(x, y)dy$

GPR Motivation

So far: Linear Function plus Gaussian Noise

Recap: We ...

- assumed a linear model with normally distributed noise $\mathcal{N}(0, \sigma^2)$
- also assumed **uniform priors**
- used **maximum likelihood estimation** to find best model
- assumed independence of the data so that we could rewrite the likelihood as a product of the individual likelihoods
- used **linear least squares** (minimize SSE_w)
- used **gradient descent** or **normal equations**
- used projection functions ϕ
- added **regularization**

Possibility of improvement

But we did not consider learning the confidence of our model.

Intuition:

- Where we have data our model should be secure:
  - data without noise: 100% confidence
  - data with noise: still some uncertainty
- Where we do **not** have data our model should be uncertain
- The further away from data points the less confident our model should be

Intuitive GPR derivation

- Let us assume still a linear model with normally distributed noise $p(y_i|x_i, w) \sim \mathcal{N}(w^T x, \sigma^2 I)$

What we want is maximum likelihood (MLE) of our model w given the data D: $p(D|w)$

This can be written as the product of the individual likelihoods per data point: $p(y_i|x_i, w)$

Hence:

$$p(D|w) = \prod_i^n p(y_i|x_i, w)$$

$\Rightarrow$ This means that $p(D|w)$ is also gaussian distributed.

Intuitive GPR derivation

- Why model $p(w|D)$ and make assumptions about the model w?
- We also could model $p(y|x, D)$ directly.

$$p(y|x, D) = \int_w p(y|x, D, w)dw = \int_w p(y|x, w)p(w|D)dw$$

According to Bayes Theorem:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

- Assumption: prior of the model w is Gaussian distributed.

$\Rightarrow p(y|x, D)$ is Gaussian distributed $p(y|x, D) \sim \mathcal{N}(\mu, \Sigma)$

GPR - using Kernels

$$p(y|x, D) \sim \mathcal{N}(\mu, \Sigma)$$

- We want that similar data points have a high covariance
- We can construct the corvariance matrix using Kernels
- $\Sigma = K$ with $K_{ij} = k(x_i, x_j)$

Conditional distribution

$$\text{Given: } \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

we can derive the conditional distribution as (Theorem):

$$\mu_2|\mu_1 = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1)$$
$$\Sigma_2|\Sigma_1 = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$$

- Using the covariance matrix of our data we can use this as a kernel K.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \right)$$

Correlating data and prediction

We want to correlate now our data and the prediction for our new data.

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$$

Using the theorem for conditional distributions we get:

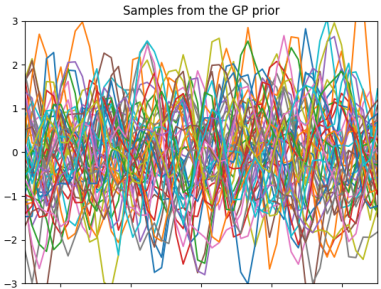
$$\begin{aligned} y_*(x) = \mu_* &= K_*^T K^{-1} f \\ var(y_*) = \Sigma_* &= K_{**} - K_*^T K^{-1} K_* \end{aligned}$$

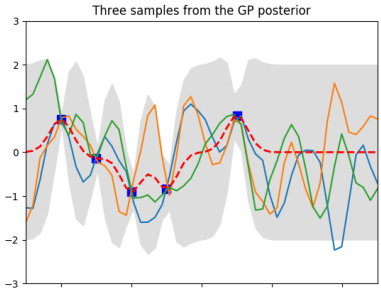
To get the inverse of K one can use the Cholesky decomposition.

Function Space View

- Consider the space of all available functions.

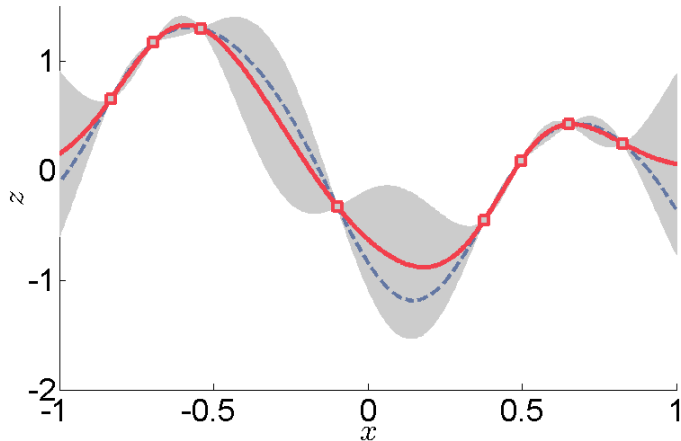
$\rightarrow$ Reduce the space such that the functions fit to the data and describe the resulting new space of functions.





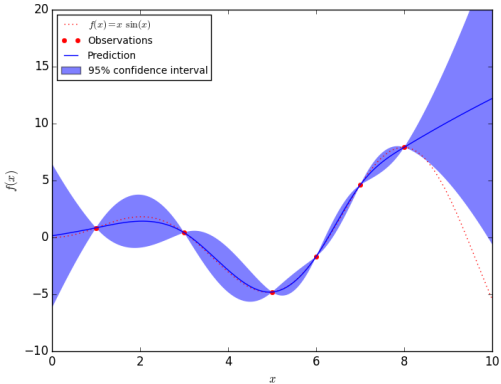
Scripts used from `http://katbailey.github.io/post/gaussian-processes-for-dummies/`

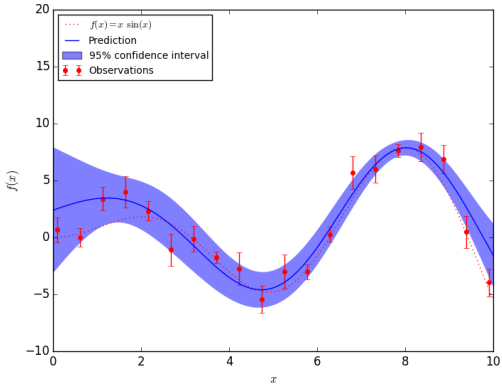
Example 1: Gaussian Process vs. Spline



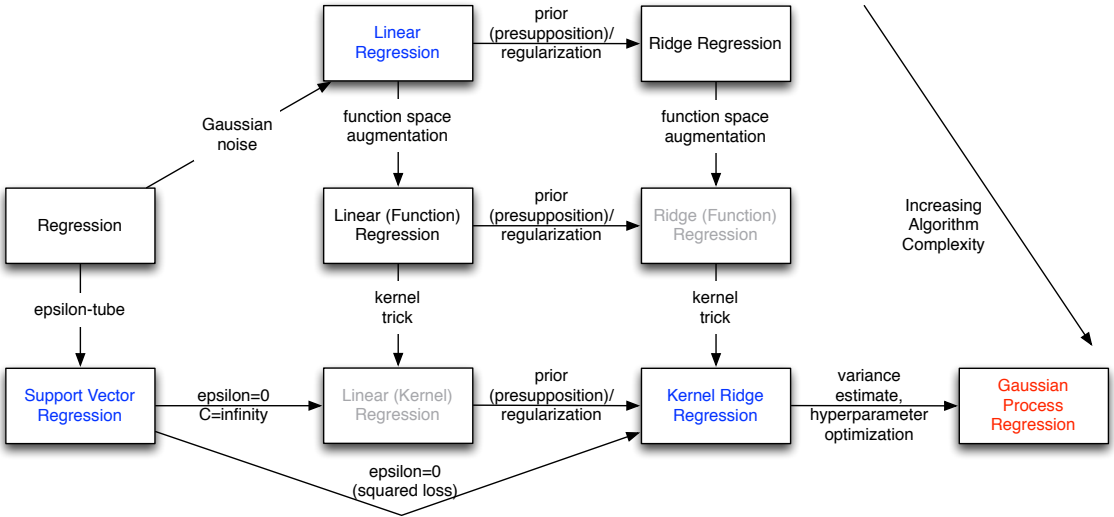
"Example of kriging interpolation in 1D" by Antro5 - Produced using the Small (Matlab/GNU Octave) Toolbox for Kriging STK. Licensed under CC BY-SA 3.0 via Wikipedia

Example 2: Clean vs. Noisy Data (scikit-learn)





Overall Scheme of Algorithm Connections



Algorithm Class Perspective

Algorithm Class Perspective

An "algorithm" in ML is NOT a single implementation but usually a selection of possible ingredients from different categories – it is more a class or category:

- Function space (Which type of functions are modeled in this class?)
- Objective function components (target function/error function, regularization term)
- Implementation (How do I get a solution of the optimization problem?)

Function Space

- Linear
- Model arbitrary functions with feature augmentation
- Kernel function
- Sum of gaussians (Gaussian mixture models)
- locally defined functions (e.g., locally weighted linear regression)
- Neural network functions

Objective for Optimization

- Regularization/prior (none/uniform, 2-norm/Gaussian, 1-norm/Laplacian)
- Loss/error term (squared, absolute values, hinge loss, ϵ-insensitive loss)
- Likelihood (often sum of squared errors)
- Sum of squared errors
- Distance to neighbor
- Cross entropy
- Sometimes not directly defined (e.g., neural network with dropout or DBSCAN)

Implementations

- "Explicit" solution formula (e.g., in linear regression)
- Different eigenvalue solvers (e.g., for PCA)
- Gradient descent, Coordinate descent
- Stochastic gradient descent (gradient descent on sample level)
- Batch wise optimization: 1, 2, subgroup, (e.g., SVM solvers)
- second order methods like BFGS (Broyden-Fletcher-Goldfarb-Shanno algorithm)
- Expectation maximization (EM, e.g., as in GMMs)
- Two loops around different parts of parameter space
- Hyperparameter optimization (e.g., DBSCAN heuristics, leave-one-out-crossvalidation, maximum likelihood, and BFGS for Gaussian process regression hyperparameters)

Sources

- http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html

Thank You!
Please feel free to ask questions in the forums.