Kernel Methods for Regression

Support Vector Regression
Gaussian Mixture Regression
Gaussian Process Regression
Problem Statement

Predict $y$ given input $x$ through a non-linear function $f$:

$$y = f(x)$$

Estimate $f$ that best predict set of training points $\{x^i, y^i\}_{i=1,...,M}$?
Non-linear regression and the Kernel Trick

Non-Linear regression: Fit data with a function that is not linear in the parameters

\[ y = f(x; \alpha); \quad \alpha: \text{parameters of the function} \]

Non-parametric regression: use the data to determine the parameters of the function so that the problem can be again phrased as a linear regression problem.

**Kernel Trick:** Send data in feature space with non-linear function and perform linear regression in feature space

\[ y = \sum_i \alpha_i k(x^i, x) \]

\( x^i \): datapoints, k: kernel fct.
Data-driven Regression

Good prediction depends on the choice of datapoints.
Data-driven Regression

Good prediction depends on the choice of datapoints.

The more datapoints, the better the fit.

Computational costs increase dramatically with number of datapoints.
Kernel methods for regression

Several methods in ML for performing non-linear regression. Differ in the objective function, in the amount of parameters.

*Gaussian Process Regression (GPR) uses all datapoints*
Kernel methods for regression

Several methods in ML for performing non-linear regression.
Differ in the objective function, in the amount of parameters.

*Gaussian Process Regression (GPR) uses all datapoints*
*Support Vector Regression (SVR) picks a subset of datapoints (support vectors)*
Kernel methods for regression

Several methods in ML for performing non-linear regression.
Differ in the objective function, in the amount of parameters.

*Gaussian Process Regression (GPR) uses all datapoints*
*Support Vector Regression (SVR) picks a subset of datapoints (support vectors)*
*Gaussian Mixture Regression (GMR) generates a new set of datapoints (centers of Gaussian functions)*

Blue: true function
Red: estimated function
Kernel methods for regression

**Deterministic regressive model**

\[ y = f(x), \quad x \in \mathbb{R}^N, \quad y \in \mathbb{R} \]

**Probabilistic regressive model**

\[ y = f(x) + \epsilon, \quad \text{with } \epsilon = N(0, \sigma^2) \]

Build an estimate of the noise model and then compute \( f \) directly (Support Vector Regression)
Support Vector Regression
Support Vector Regression (SVR)

Assume a nonlinear mapping $f$, s.t. $y = f(x)$.

How to estimate $f$ to best predict the pair of training points $\{x^i, y^i\}_{i=1,...,M}$?

How to generalize the support vector machine framework for classification to estimate continuous functions?

1. Assume a non-linear mapping through feature space and then perform linear regression in feature space

2. Supervised learning – minimizes an error function.

→ First determine a way to measure error on testing set in the linear case!
Support Vector Regression

Assume a linear mapping \( f \), s.t. \( y = f(x) = w^T x + b \).

Measure the error on prediction

How to estimate \( w \) and \( b \) to best predict the pair of training points \( \{x^i, y^i\}_{i=1,...,M} \)?

\( b \) is estimated as in SVR through least-square regression on support vectors; hence we omit it from the rest of the developments.
Support Vector Regression

Set an upper bound on the error $\varepsilon$ and consider as correctly classified all points such that $f(x) - y \leq \varepsilon$.

Penalize only datapoints that are not contained in the $\varepsilon$-tube.
Support Vector Regression

The $\varepsilon$-margin is a measure of the width of the $\varepsilon$-insensitive tube and hence of the precision of the regression.

A small $||w||$ corresponds to a small slope for $f$. In the linear case, $f$ is more horizontal.
Support Vector Regression

A large $||w||$ corresponds to a large slope for $f$. In the linear case, $f$ is more vertical.

The flatter the slope of the function $f$, the larger the $\varepsilon$–margin.

→ To maximize the margin, we must minimize the norm of $w$. 

$y = wx + b$
Support Vector Regression

This can be rephrased as a constraint-based optimization problem of the form:

\[
\begin{align*}
\text{minimize } & \frac{1}{2} \|w\|^2 \\
\text{subject to } & \begin{cases}
\langle w, x^i \rangle + b - y^i \leq \varepsilon \\
y^i - \langle w, x^i \rangle - b \leq \varepsilon
\end{cases}
\end{align*}
\]

\[\forall i = 1, \ldots, M\]

Need to penalize points outside the \(\varepsilon\)-insensitive tube.
Support Vector Regression

Introduce slack variables $\xi_i, \xi_i^*, C \geq 0$:

$$\frac{1}{2} \|w\|^2 + \frac{C}{M} \sum_{i=1}^{M} (\xi_i + \xi_i^*)$$

subject to

$$\begin{align*}
\langle w, x^i \rangle + b - y^i & \leq \varepsilon + \xi_i \\
y^i - \langle w, x^i \rangle - b & \leq \varepsilon + \xi_i^* \\
\xi_i & \geq 0, \quad \xi_i^* \geq 0
\end{align*}$$

Need to penalize points outside the $\varepsilon$-insensitive tube.
Support Vector Regression

Introduce slack variables $\xi_i, \xi_i^*, C \geq 0$:

\[
\begin{aligned}
&\text{minimize } \frac{1}{2} \|w\|^2 + \frac{C}{M} \sum_{i=1}^{M} (\xi_i + \xi_i^*) \\
&\text{subject to } \begin{cases}
\langle w, x^i \rangle + b - y^i \leq \varepsilon + \xi_i \\
y^i - \langle w, x^i \rangle - b \leq \varepsilon + \xi_i^* \\
\xi_i \geq 0, \quad \xi_i^* \geq 0
\end{cases}
\end{aligned}
\]

We now have the solution to the linear regression problem.

How to generalize this to the nonlinear case?
Support Vector Regression

Lift $x$ into feature space and then perform linear regression in feature space.

Linear Case:
\[ y = f(x) = \langle w, x \rangle + b \]

Non-Linear Case:
\[ x \rightarrow \phi(x) \]
\[ y = f(\phi(x)) = \langle w, \phi(x) \rangle + b \]

\[ w \text{ lives in feature space!} \]
Support Vector Regression

In feature space, we obtain the same constrained optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + \frac{C}{M} \sum_{i=1}^{M} (\xi_i + \xi_i^*) \\
\text{subject to} & \quad \begin{cases} \\
\langle w, \phi(x^i) \rangle + b - y^i \leq \varepsilon + \xi_i \\
y^i - \langle w, \phi(x^i) \rangle - b \leq \varepsilon + \xi_i^* \\
\xi_i \geq 0, \quad \xi_i^* \geq 0
\end{cases}
\end{align*}
\]
Support Vector Regression

Again, we can solve this quadratic problem by introducing sets of Lagrange multipliers and writing the Lagrangian:

\[
L(w, \xi, \xi^*, b) = \frac{1}{2}\|w\|^2 + \frac{C}{M} \sum_{i=1}^{M} (\xi_i + \xi_i^*) - \frac{C}{M} \sum_{i=1}^{M} (\eta_i \xi_i + \eta_i^* \xi_i^*) \\
- \sum_{i=1}^{M} \alpha_i \left( \varepsilon + \xi_i + y_i - \left<w, \phi(x^i)\right> - b \right) \\
- \sum_{i=1}^{M} \alpha_i^* \left( \varepsilon + \xi_i^* - y_i + \left<w, \phi(x^i)\right> + b \right)
\]
Support Vector Regression

Requiring that the partial derivatives are all zero

\[
\frac{\partial L}{\partial b} = \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) = 0;
\]

\[
\frac{\partial L}{\partial w} = w - \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) \phi(x^i) = 0;
\]

\[
\frac{\partial L}{\partial \xi_i^*(i)} = \frac{C}{M} - \alpha_i^* - \eta_i^* = 0
\]

And replacing in the primal Lagrangian, we get the Dual optimization problem:

\[
\begin{aligned}
\max_{\alpha, \alpha^*} & \quad \left\{- \frac{1}{2} \sum_{i, j=1}^{M} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) \cdot k(x^i, x^j) \right. \\
& \left. - \varepsilon \sum_{i=1}^{M} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{M} y^i (\alpha_i^* + \alpha_i) \right\}
\end{aligned}
\]

subject to \( \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) = 0 \) and \( \alpha_i^*, \alpha_i \in \left[ 0, \frac{C}{M} \right] \)
Support Vector Regression

The solution is given by:

\[ y = f(x) = \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) k(x^i, x) + b \]

- Linear Coefficients (Lagrange multipliers for each constraint).
- If Gaussian Kernel, \( M \) Gaussians centered on each training datapoint.
Support Vector Regression

The solution is given by:

\[ y = f(x) = \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) k(x^i, x) + b \]

Kernel places a Gauss function on each SV
Support Vector Regression

The solution is given by:

\[ y = f(x) = \sum_{i=1}^{M} (\alpha_i - \alpha^*_i) k(x^i, x) + b \]

The Lagrange multipliers define the importance of each Gaussian function.

Converges to \( b \) when SV effect vanished.

\[ \alpha_1^* = 1.5 \quad \alpha_2 = 2 \quad \alpha_3^* = 1.5 \quad \alpha_4 = 3 \quad \alpha_5^* = 1 \quad \alpha_6 = 2.5 \]
\( \varepsilon \)-SVR: Hyperparameters

The solution to SVR we just saw is referred to as \( \varepsilon \)-SVR

Two Hyperparameters

minimize \( \frac{1}{2} \| w \|^2 + \frac{C}{M} \sum_{i=1}^{M} (\xi_i + \xi_i^*) \)

subject to

\[
\begin{align*}
\langle w, x^i \rangle + b - y^i &\leq \varepsilon - \xi_i \\
y^i - \langle w, x^i \rangle - b &\leq \varepsilon + \xi^*_i \\
\xi_i &\geq 0, \quad \xi_i^* \geq 0
\end{align*}
\]

C controls the penalty term on poor fit
\( \varepsilon \) determines the minimal required precision
\( \varepsilon \)-SVR: Effect of Hyperparameters

Effect of the RBF kernel width on the fit. Here fit using \( C=1000, \varepsilon=0.01, \) kernel width=0.1.
$\varepsilon$-SVR: Effect of Hyperparameters

Effect of the RBF kernel width on the fit. Here fit using $C=1000$, $\varepsilon=0.01$, kernel width=0.01 $\rightarrow$ Overfitting
\( \varepsilon \)-SVR: Effect of Hyperparameters

Effect of the RBF kernel width on the fit. Here fit using \( C=100, \varepsilon=0.03, \) kernel width=0.1. Reduction of the effect of the kernel width on the fit by choosing appropriate hyperparameters.
Support Vector Regression: $\nu$-SVR

As the number of data grows, so does the number of support vectors.

Introduce a new parameter as in SVM:

$$\nu \in [0, 1] \quad \min_{w, \xi, \rho} \left( \frac{1}{2} \|w\|^2 + \nu \varepsilon + \sum_{j=1}^{M} (\xi_j + \xi_j^*) \right)$$

$n$-SVR puts an upper bound on the support vectors

It allows to fit automatically the epsilon-tube!
Support Vector Regression: \( \nu\text{-SVR} \)

Effect of the automatic adaptation of \( \varepsilon \) using \( \nu\text{-SVR} \)
Support Vector Regression: $\nu$-SVR

Effect of the automatic adaptation of $\varepsilon$ using $\nu$-SVR
Kernel methods for regression

**Deterministic regressive model**

\[ y = f(x), \quad x \in \mathbb{R}^N, \quad y \in \mathbb{R} \]

**Probabilistic regressive model**

\[ y = f(x) + \varepsilon, \quad \text{with } \varepsilon = N\left(0, \sigma^2\right) \]

Build an estimate of the noise model and then compute \( f \) directly (Support Vector Regression)
Kernel methods for regression

**Deterministic regressive model**

\[ y = f(x), \quad x \in \mathbb{R}^N, \quad y \in \mathbb{R} \]

**Probabilistic regressive model**

\[ y = f(x) + \varepsilon, \quad \text{with } \varepsilon = N(0, \sigma^2) \]

Probabilistic estimate of the nonlinear relationship between \( y \) and \( x \) through the conditional density:

\[ p(y | x) \]

(estimates the noise model and \( f \))

And then compute the estimate by taking the expectation over the conditional density:

\[
\begin{align*}
y &= E \{ f(x) + \varepsilon \} = f(x) \\
y &= E \{ p(y | x) \}
\end{align*}
\]
Gaussian Mixture Regression (GMR)

1) Estimate the joint density, $p(x,y)$, across pairs of datapoints using GMM.

$$p(x,y) = \sum_{i=1}^{K} \alpha_i \cdot p(x,y; \mu^i, \Sigma^i),$$

with $p(x,y; \mu^i, \Sigma^i) = N(\mu^i, \Sigma^i)$

$\mu^i, \Sigma^i$: mean and covariance matrix of Gaussian $i$
Gaussian Mixture Regression (GMR)

1) Estimate the *joint density*, $p(x,y)$, across pairs of datapoints *using GMM*.

$$p(x,y) = \sum_{i=1}^{K} \alpha_i \cdot p(x,y; \mu^i, \Sigma^i), \quad \text{with } p(x,y; \mu^i, \Sigma^i) = N(\mu^i, \Sigma^i)$$

$\mu^i, \Sigma^i$: mean and covariance matrix of Gaussian $i$

Parameters are learned through Expectation-maximization. Iterative procedure. Start with random initialization.
Gaussian Mixture Regression (GMR)

1) Estimate the \textit{joint density}, $p(x,y)$, across pairs of datapoints \textit{using GMM}.

$$p(x,y) = \sum_{i=1}^{K} \alpha_i p(x,y; \mu^i, \Sigma^i),$$  \hspace{1cm} \text{with} \quad p(x,y; \mu^i, \Sigma^i) = N(\mu^i, \Sigma^i)$$

$\mu^i, \Sigma^i$ : mean and covariance matrix of Gaussian $i$

Mixing Coefficients $\sum_{i=1}^{K} \alpha_i = 1$

Probability that all $M$ datapoints were generated by Gaussian $i$: \hspace{1cm}$\alpha_i = p(i) = \sum_{j=1}^{M} p(i \mid x^j)$
Gaussian Mixture Regression (GMR)

1) Estimate the *joint density*, \( p(x,y) \), across pairs of datapoints using GMM.

2) Compute the regressive signal, by taking \( p(y|x) \)

\[
p(y|x) = \sum_{i=1}^{K} \beta_i(x) \cdot p(y|x; \mu^i, \Sigma^i)
\]

with
\[
\beta_i(x) = \frac{\alpha_i p(x; \mu^i, \Sigma^i)}{\sum_{j=1}^{K} \alpha_j p(x; \mu^j, \Sigma^j)}
\]

The variance changes depending on the query point
Gaussian Mixture Regression (GMR)

1) Estimate the *joint density*, \( p(x,y) \), across pairs of datapoints *using GMM*.

2) Compute the regressive signal, by taking \( p(y|x) \)

\[
p(y|x) = \sum_{i=1}^{K} \beta_i(x) \cdot p(y|x; \mu, \Sigma) \quad \text{with} \quad \beta_i(x) = \frac{\alpha_i p(x; \mu^i, \Sigma^i)}{\sum_{j=1}^{K} \alpha_j p(x; \mu^j, \Sigma^j)}
\]

Influence of each marginal is modulated by \( \alpha \)

The factors \( \beta \) give a measure of the relative importance of each \( K \) regressive model. They are computed at each query point \( \rightarrow \) weighted regression
Gaussian Mixture Regression (GMR)

3) The regressive signal is then obtained by computing $E\{p(y|x)\}$:

$$E\{p(y|x)\} = \sum_{i=1}^{K} \beta_i(x) \cdot \left( \mu_i + \sum_{i}^{'} \left( \sum_{xx}^{'} \right)^{-1} (x - \mu_i) \right)$$

Linear combination of $K$ local regressive models
Gaussian Mixture Regression (GMR)

Computing the variance $\text{var}\{p(x,y)\}$ provides information on the uncertainty of the prediction computed from the conditional distribution.

$$\text{var}\{p(y|x)\} = \sum_{i=1}^{K} \beta_i(x) \cdot \left( (\tilde{\mu}^i(x))^2 + (\tilde{\Sigma}^i)^2 \right) - \sum_{i=1}^{K} (\beta_i(x) \cdot \tilde{\mu}^i(x))^2$$

Careful: This is not the uncertainty of the model. Use the likelihood to compute the uncertainty of the predictor!
Gaussian Mixture Regression (GMR)

Computing the variance $\text{var}\{p(x,y)\}$ provides information on the uncertainty of the prediction computed from the conditional distribution.
GMR: Sensitivity to Choice of $K$ and Initialization
GMR: Sensitivity to Choice of $K$ and Initialization

Fit with 4 Gaussians
Uniform initialization
GMR: Sensitivity to Choice of $K$ and Initialization

Fit with 4 Gaussians
Random initialization
GMR: Sensitivity to Choice of $K$ and Initialization

Fit with 10 Gaussians
Random initialization
Kernel methods for regression

**Deterministic** regressive model

\[ y = f(x), \quad x \in \mathbb{R}^N, \quad y \in \mathbb{R} \]

Probabilistic regressive model

\[ y = f(x) + \varepsilon, \quad \text{with} \quad \varepsilon = N(0, \sigma^2) \]

Probabilistic estimate of the nonlinear relationship between \( y \) and \( x \) through the conditional density:

\[ p(y \mid x) \quad \text{(estimates the noise model and } f) \]

And then compute the estimate by taking the expectation over the conditional density:

\[ y = E \{ f(x) + \varepsilon \} = f(x) \]

\[ y = E \{ p(y \mid x) \} \]

Gaussian Mixture Regression (GMR) computes first \( p(x, y) \) and, then, derives \( p(y \mid x) \).

Gaussian Process Regression (GPR) computes directly \( p(y \mid x) \).
Probability Density Function and Regression

A signal $y$ can be estimated through regression $y = f(x)$ by taking the expectation over the conditional probability of $p$ on $x$, for a choice of parameters for $p$:

$$y = E\{p(y | x)\}$$

The simplest way to estimate $p(y|x)$ is through Probabilistic Regression that estimates a linear regressive model.
Probabilistic Regression (PR)

PR is a statistical approach to classical linear regression that estimates the relationship between zero-mean variables $y$ and $x$ by building a linear model of the form:

$$y = f(x, w) = w^T x, \quad w, x \in \mathbb{R}^N$$

If one assumes that the observed values of $y$ differ from $f(x)$ by an additive noise $\varepsilon$ that follows a zero-mean Gaussian distribution (such an assumption consists of putting a *prior distribution* over the noise), then:

$$y = w^T x + \varepsilon, \quad \text{with } \varepsilon = N\left(0, \sigma^2\right)$$
**Probabilistic Regression**

Training set of $M$ pairs of data points \( \{X, y\} = \{x^i, y^i\}_{i=1}^M \)

Likelihood of the regressive model

\[
y = w^T X + N(0, \sigma^2)
\]

\[
\Rightarrow y \sim p(y | X, w, \sigma)
\]

Data points are independently and identically distributed (i.i.d)

\[
p(y | X, w, \sigma) \sim \prod_{i=1}^{M} p(y^i | x^i, w, \sigma)
\]

\[
= \prod_{i=1}^{M} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y^i - w^T x^i)^2}{2\sigma^2}\right)
\]
Probabilistic Regression

Training set of $M$ pairs of data points $\{X, y\} = \{x^i, y^i\}_{i=1}^M$

Likelihood of the regressive model

$$y = w^T X + N\left(0, \sigma^2\right)$$

$$\Rightarrow y \sim p(y | X, w, \sigma)$$

Prior model on distribution of parameter $w$:

$$p(w) = N\left(0, \Sigma_w\right) = \exp\left(-\frac{1}{2} w^T \Sigma_w^{-1} w\right)$$

Hyperparameters
Given by user
Probabilistic Regression

\[ p(y | x, X, y) = N \left( \frac{1}{\sigma^2} x^T A^{-1} X y, x^T A^{-1} x \right) \]

with \( A = \frac{1}{\sigma^2} \) \( XX^T + \Sigma_w^{-1} \).
From Probabilistic Regression to Gaussian Process Regression

How to extend the simple linear Bayesian regressive model for nonlinear regression, such that the non-linear problem becomes linear again?

\[ y = w^T x + N(0, \sigma^2) \]

\[ y = w^T \phi(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2) \]
Gaussian Process Regression

How to extend the simple linear Bayesian regressive model for nonlinear regression, such that the non-linear problem becomes linear again?

\[ y = w^T x + N\left(0, \sigma^2\right) \]

\[
p(y | x, X, y) = N\left(\frac{1}{\sigma^2} x^T A^{-1} X y, x^T A^{-1} x\right), \quad A = \frac{1}{\sigma^2} XX^T + \Sigma_w^{-1}
\]

\[ \phi(x) \]

Non-Linear Transformation

\[
p(y | x, X, y) = N\left(\frac{1}{\sigma^2} \phi(x)^T A^{-1} \Phi(X) y, \phi(x)^T A^{-1} \phi(x)\right)
\]

with \[ A = \sigma^{-2} \Phi(X) \Phi(X)^T + \Sigma_w^{-1} \]
Gaussian Process Regression

\[ y = E \{ y \mid x, X, y \} = \sum_{i=1}^{M} \alpha_i k(x, x^i) \]

with \( \alpha = \left[ K(X, X) + \sigma^2 I \right]^{-1} y \)  

Take as kernel  \( k(x, x') = \phi(x)^T \Sigma_w \phi(x') \)  

Inner product in feature space

\[ p(y \mid x, X, y) = N \left( \frac{1}{\sigma^2} \phi(x)^T A^{-1} \Phi(X)y, \phi(x)^T A^{-1} \phi(x) \right) \]

with \( A = \sigma^{-2} \Phi(X) \Phi(X)^T + \Sigma_w^{-1} \)
Gaussian Process Regression

\[
y = E \{ y \mid x, X, y \} = \sum_{i=1}^{M} \alpha_i k(x, x^i)
\]

with \( \alpha = \left[ K(X, X) + \sigma^2 I \right]^{-1} y \)

\( \alpha > 0 \) 

\( \rightarrow \) All datapoints are used in the computation!
Gaussian Process Regression

\[ y = E \{ y \mid x, X, y \} = \sum_{i=1}^{M} \alpha_i k(x, x^i) \]

with \( \alpha = \left[ K(X, X) + \sigma^2 I \right]^{-1} y \)

The kernel and its hyperparameters are given by the user. These can be optimized through maximum likelihood over the marginal likelihood, i.e. \( p(y|X; \text{parameters}) \).
Gaussian Process Regression

Sensitivity to the choice of *kernel width* (called *lengthscale* in most books) when using Gaussian kernels (also called RBF or square exponential).

$$k(x, x') = e^{-\|x-x'\|}$$

Kernel Width=0.1
Gaussian Process Regression

Sensitivity to the choice of kernel width (called lengthscale in most books) when using Gaussian kernels (also called RBF or square exponential).

\[ k(x, x') = e^{-\frac{\|x-x'\|}{l}} \]

Kernel Width=0.5
Gaussian Process Regression

\[ y = E \{ y \mid x, X, y \} = \sum_{i=1}^{M} \alpha_i k(x, x^i) \]

with \( \alpha = \left[ K(X, X) + \sigma^2 I \right]^{-1} y \)

The value for the noise needs to pre-set by hand.

\[ \text{cov} \left( p(y \mid x) \right) = K(x, x) - K(x, X) \left[ K(X, X) + \sigma^2 I \right]^{-1} K(X, x) \]

The larger the noise, the more uncertainty. The noise is \( \leq 1 \).
Gaussian Process Regression

Low noise: $\sigma=0.05$
Gaussian Process Regression

High noise: $\sigma=0.2$
Comparison Across Methods

Generalization – prediction away from datapoints

\[ y = E \{ y \mid x, X, y \} = \sum_{i=1}^{M} \alpha_i k(x, x^i) \]

GPR Predict \( y=0 \) away from datapoints!

SVR predicts \( y=b \) away from datapoints

\[ y = \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) k(x^i, x) + b \]
Comparison Across Methods

Generalization – prediction away from datapoints

GMR predicts the trend away from data
Comparison Across Methods

Generalization – prediction away from datapoints

But prediction depends on initialization and solution found during training
Variance in $p(y|x)$ in GMR represents uncertainty of predictive model.

Variance in $p(y|x)$ in GPR represents uncertainty of predictive mode.
Comparison Across Methods

Variance in SVR represents the epsilon-tube and does not represent uncertainty of the model either! No measure of uncertainty in SVR!
SVR, GPR, GMR: Similarities

- SVR, GMR and GPR are based on the same regressive model:

\[ y = f(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2) \]

- GMR and GPR are Gaussian Conditional Distributions

- SVR, GMR and GPR compute a weighted combination of local predictors

- SVR, GMR and GPR Separate input space into regions modeled by Gaussian distributions (true only when using Gaussian kernels for GPR and SVR)

- GMR allows to predict multi-dimensional outputs, while SVR and GPR can predict only a uni-dimensional output y.
SVR, GPR, GMR: Differences

SVR, GMR and GPR are based on the same probabilistic regressive model. But they do not optimize the same objective function → find different solutions.

- **SVR:**
  - minimizes reconstruction error through convex optimization → ensured to find the *optimal* estimate; but not unique solution
  - usually finds a nm of models ≤ nm of datapoints (support vectors)

- **GMR:**
  - learns p(x,y) through maximum likelihood → finds *local* optimum
  - compute a generative model p(x,y) from which it derives p(y|x)
  - starts with a low nm of models ≪ nm of datapoints

- **GPR:**
  - No optimization; analytical (*optimal*) solution
  - expresses p(y|x) as a full density model
  - nm of models = nm of datapoints!
Hyperparameters of SVR, GPR, GMR

SVR, GMR and GPR all depend on hyperparameters that need to be determined beforehand. These are:

• SVR
  • choice of error margin \( \varepsilon \), which can be replaced by choice of \( \nu \) in \( \nu \)-SVM.
  • choice of kernel and associated kernel parameters

• GMR:
  • choice of the number of Gaussians
  • choice of initialization (affects convergence to local optimum)

• GPR:
  • choice of noise parameter
  • choice of kernel width (length-scale)

The hyperparameters can be optimized separately; e.g. the \( \text{nm} \) of Gaussians in GMR can be estimated using BIC; the lengthscale and noise of GPR can be estimated through maximum likelihood and the kernel parameters of SVR can be optimized through grid search.
Conclusion

No easy way to determine which regression technique fits best your problem

**Training**

- **SVR**
  - Convex optimization

- **GMR**
  - EM, iterative technique, needs several runs

- **GPR**
  - Analytical solution
  - One shot, but uses all data-points

**Testing**

- **SVR**
  - Few SV or Gaussian fct
  - Small fraction of original data

- **GMR**
  - Keeps all the data points